Exhibit 9

US005436637A

United States Patent [19]

Gayraud et al.

[11] **Patent Number: 5,436,637**

[45] **Date of Patent: Jul. 25, 1995**

[54] **GRAPHICAL USER INTERFACE SYSTEM AND METHODS FOR IMPROVED USER FEEDBACK**

[75] Inventors: **Charles E. Gayraud; Perry A. Gee,** both of Santa Cruz, Calif.

[73] Assignee: **Borland International, Inc.,** Scotts Valley, Calif.

[21] Appl. No.: **26,720**

[22] Filed: **Mar. 5, 1993**

[51] **Int. Cl.**6 .... **G09G 5/14;** G09G 5/40

[52] **U.S. Cl.** **345/116;** 345/146; 395/155; 395/159

[58] **Field of Search** 345/116, 145, 146; 395/155, 156, 159

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,310,839	1/1982	Schwerdt	345/146
4,622,013	11/1986	Cerchio	434/118
4,656,603	4/1987	Dunn	345/146
4,742,473	5/1988	Shugar et al.	395/156
4,789,962	12/1988	Berry	395/156
4,964,077	10/1990	Eisen	395/275
5,123,087	6/1992	Newell	395/155
5,235,679	8/1993	Yoshizawa et al.	395/156

OTHER PUBLICATIONS

Simpson, Alan, *Mastering WordPerfect 5.1 & 5.2 for Windows,* Sybex Publication, 1993, pp. 6–8, 69–70, 646–649, 659, 665–666.

Webster, Tony & Associates, *WordPerfect 5.1 for Windows by Example,* Webster & Associates Publication, 1992, pp. 1, 3, 9, 381, 385, 551–554.

Matthies, Kurt W. G., *Balloon Help Takes Off,* MacUser, Dec. 1991, pp. 241–248.

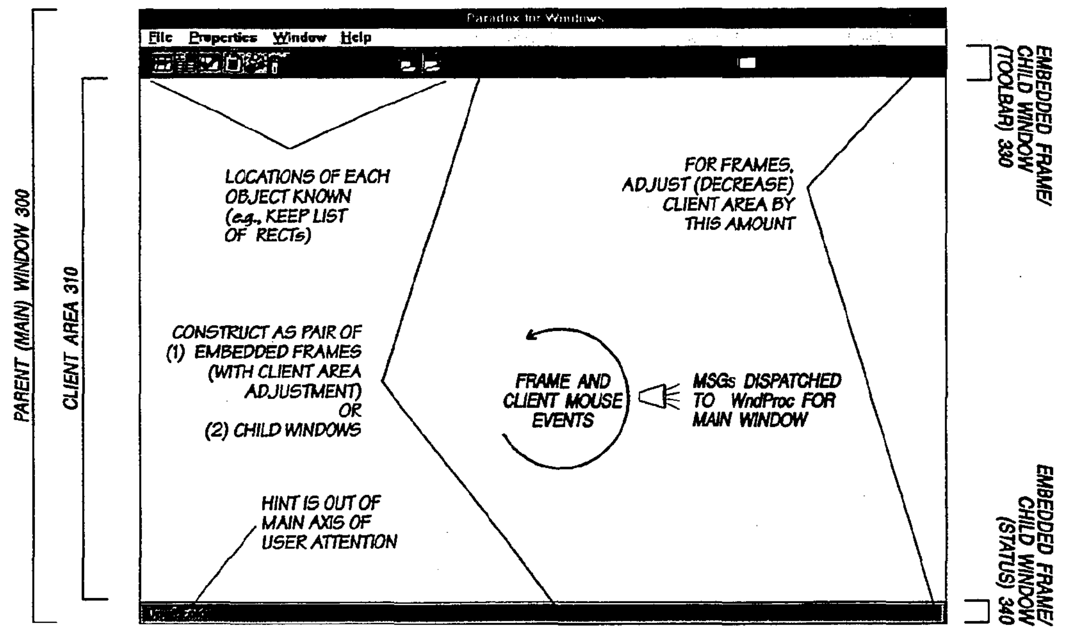
Primary Examiner—Jeffery Brier

Attorney, Agent, or Firm—John A. Smart; Vernon A. Norviel; Michael J. Ritter

[57] **ABSTRACT**

Graphical user interface system and methods for providing "hints" for screen objects of interest, particularly user interface elements which rely upon multitudes of iconic (bitmap image) indicia. The interface includes a static (status) frame or window positioned in a non-intrusive fashion below or to one side of a client area (active portion) of a user interface. The frame is continually updated with appropriate descriptors or "hints" (e.g., text and/or graphics) as a screen cursor moves from one screen object to another.

40 Claims, 19 Drawing Sheets


Paradox for Windows
File Properties Window Help
LOCATIONS OF EACH OBJECT KNOWN (e.g., KEEP LIST OF RECTs)
FOR FRAMES, ADJUST (DECREASE) CLIENT AREA BY THIS AMOUNT
CONSTRUCT AS PAIR OF (1) EMBEDDED FRAMES (WITH CLIENT AREA ADJUSTMENT) OR (2) CHILD WINDOWS
FRAME AND CLIENT MOUSE EVENTS
MSGs DISPATCHED TO WndProc FOR MAIN WINDOW
HINT IS OUT OF MAIN AXIS OF USER ATTENTION
PARENT (MAIN) WINDOW 300
CLIENT AREA 310
EMBEDDED FRAME/ CHILD WINDOW (TOOLBAR) 330
EMBEDDED FRAME/ CHILD WINDOW (STATUS) 340

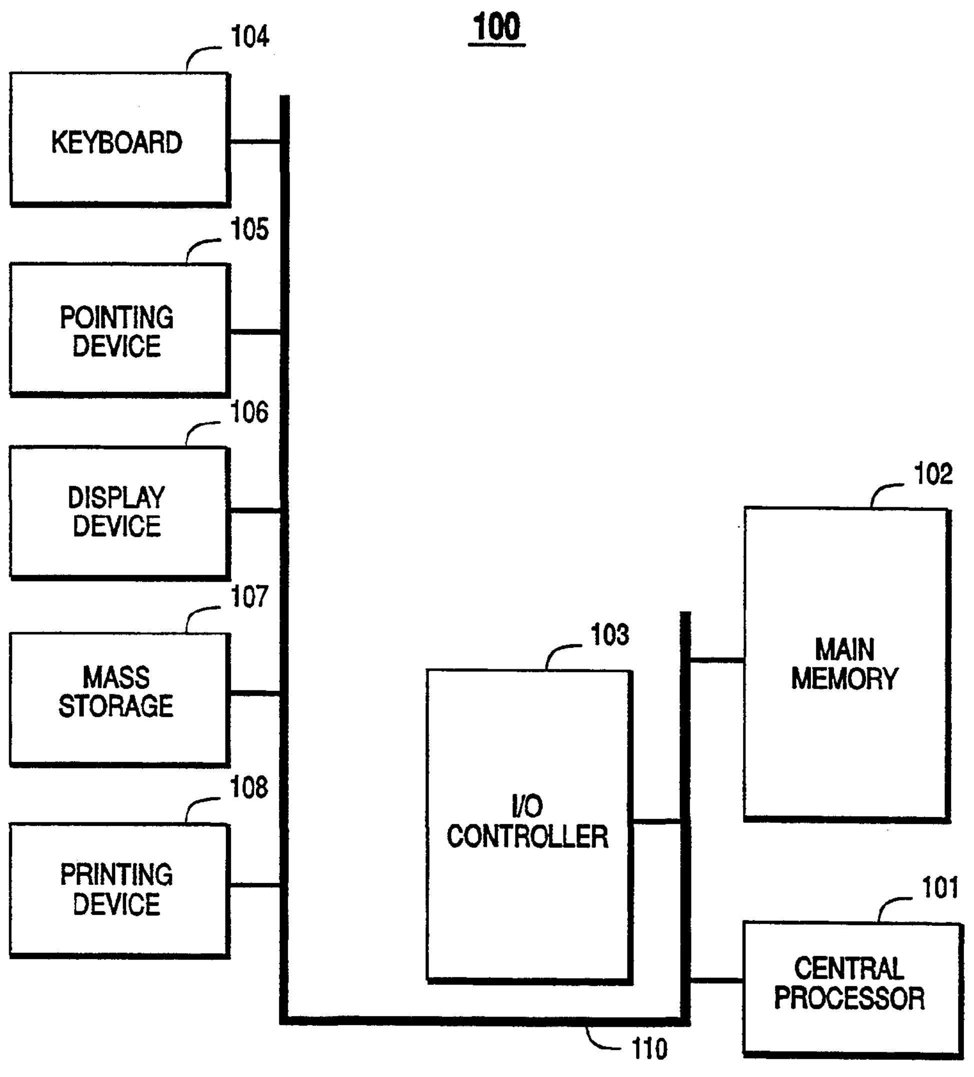

100
104
KEYBOARD
105
POINTING DEVICE
106
DISPLAY DEVICE
107
MASS STORAGE
108
PRINTING DEVICE
103
I/O CONTROLLER
102
MAIN MEMORY
101
CENTRAL PROCESSOR
110

FIG. 1A

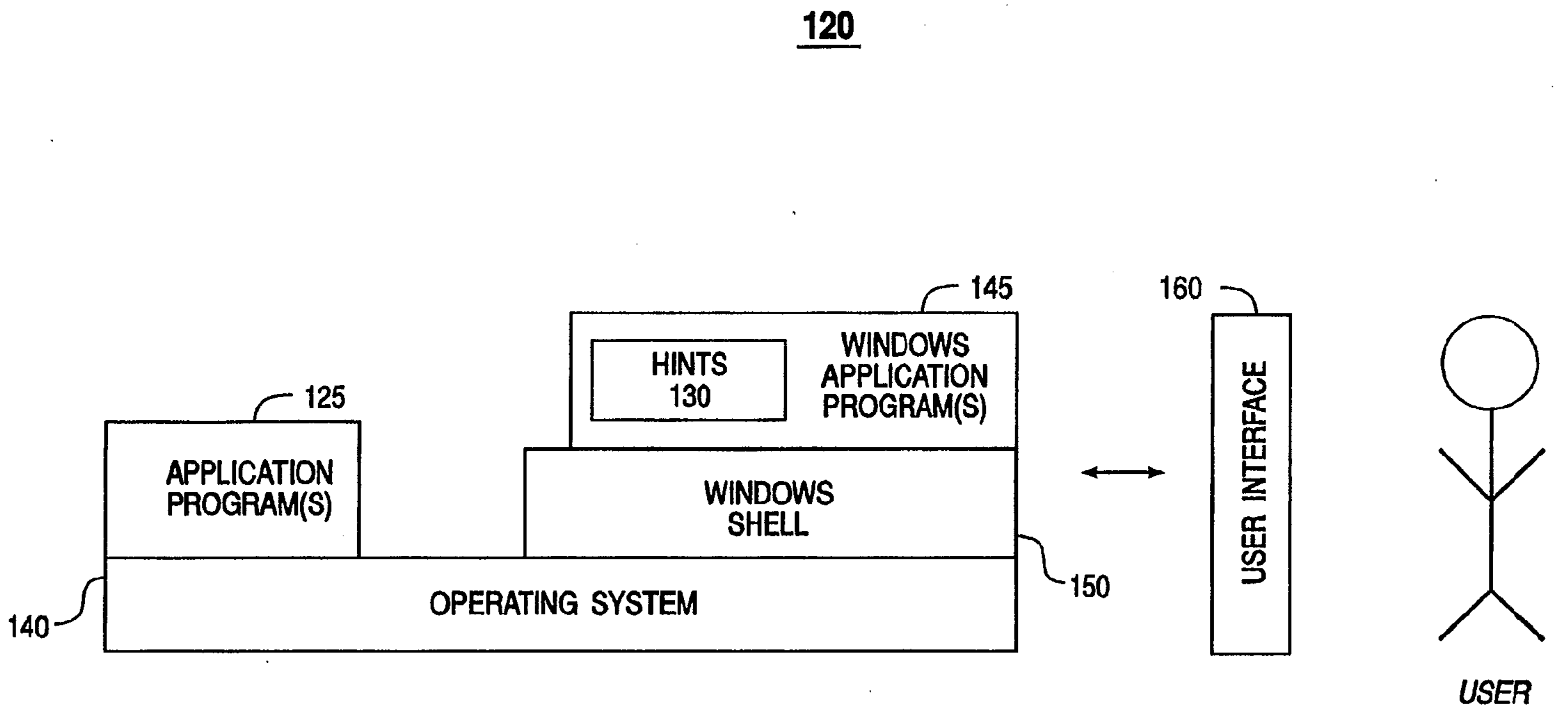

120
125
APPLICATION PROGRAM(S)
145
HINTS 130
WINDOWS APPLICATION PROGRAM(S)
WINDOWS SHELL
150
OPERATING SYSTEM
140
160
USER INTERFACE
USER

FIG. 1B

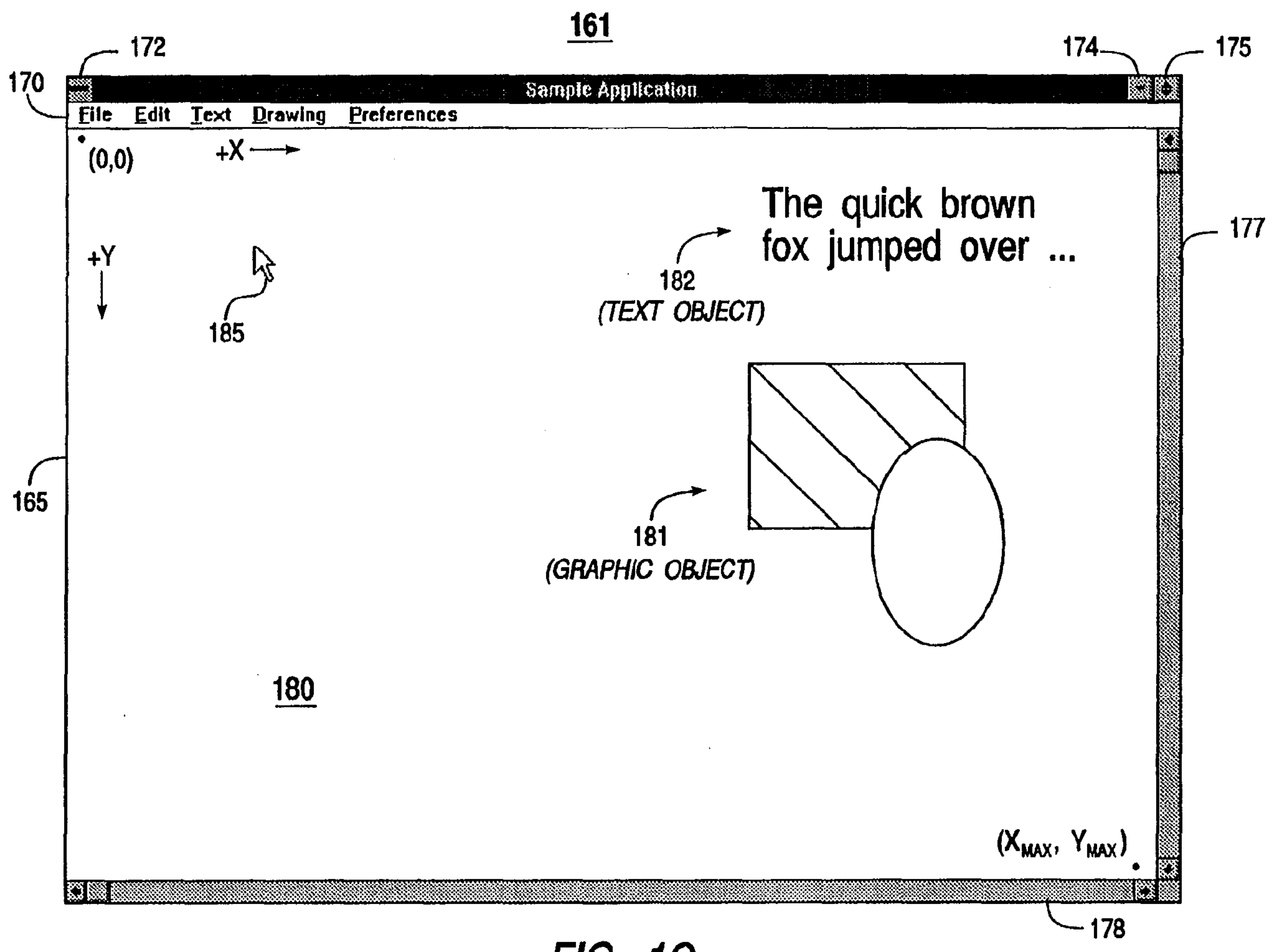

161
172
170
174
175
Sample Application
File Edit Text Drawing Preferences
(0,0)
+X
+Y
185
The quick brown fox jumped over ...
182
(TEXT OBJECT)
177
165
181
(GRAPHIC OBJECT)
180
(XMAX, YMAX)
178

FIG. 1C

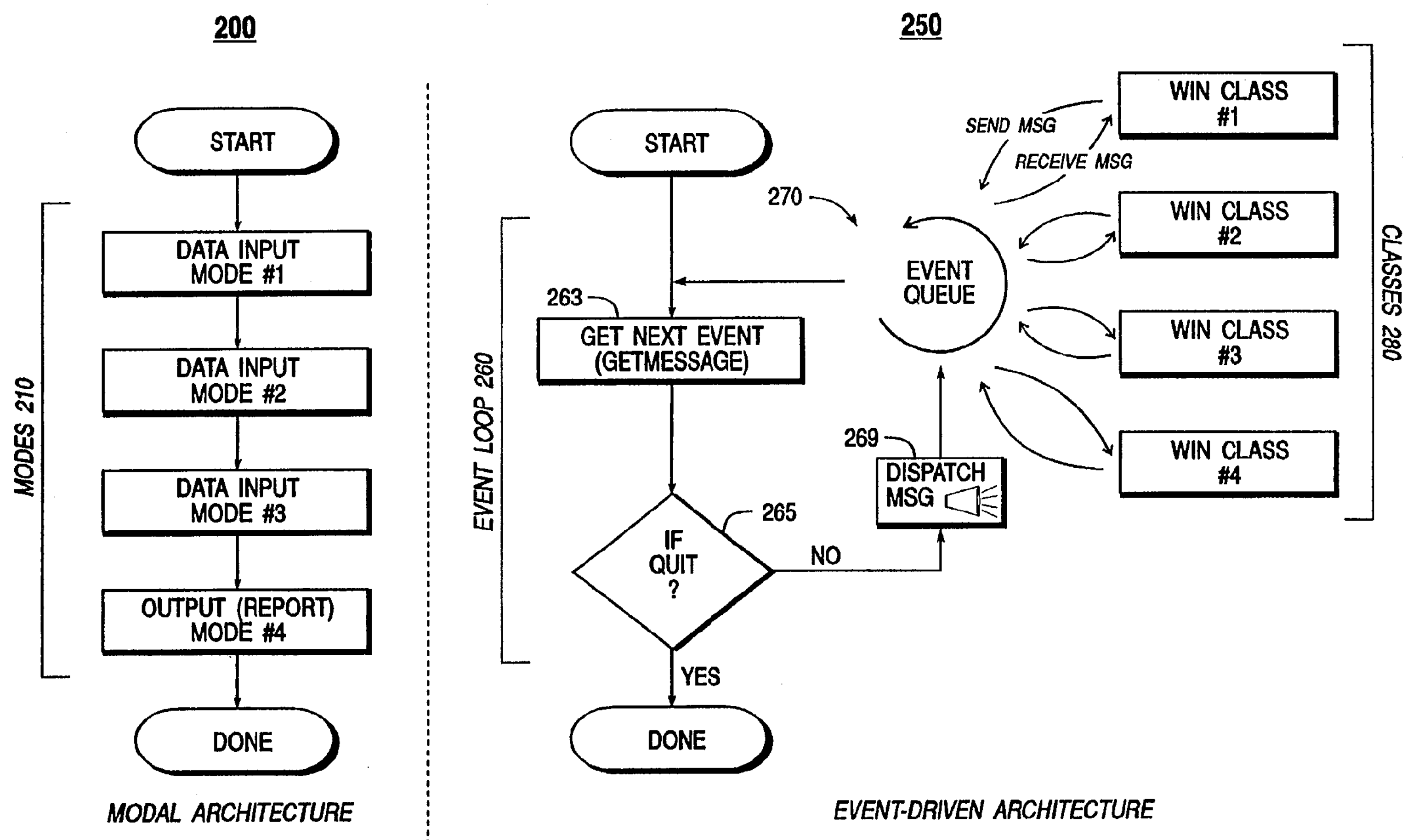

200
START
DATA INPUT MODE #1
DATA INPUT MODE #2
DATA INPUT MODE #3
OUTPUT (REPORT) MODE #4
DONE
MODES 210
MODAL ARCHITECTURE
250
START
263
GET NEXT EVENT (GETMESSAGE)
EVENT LOOP 260
IF QUIT ?
265
NO
YES
DONE
270
EVENT QUEUE
269
DISPATCH MSG
SEND MSG
RECEIVE MSG
WIN CLASS #1
WIN CLASS #2
WIN CLASS #3
WIN CLASS #4
CLASSES 280
EVENT-DRIVEN ARCHITECTURE

FIG. 2

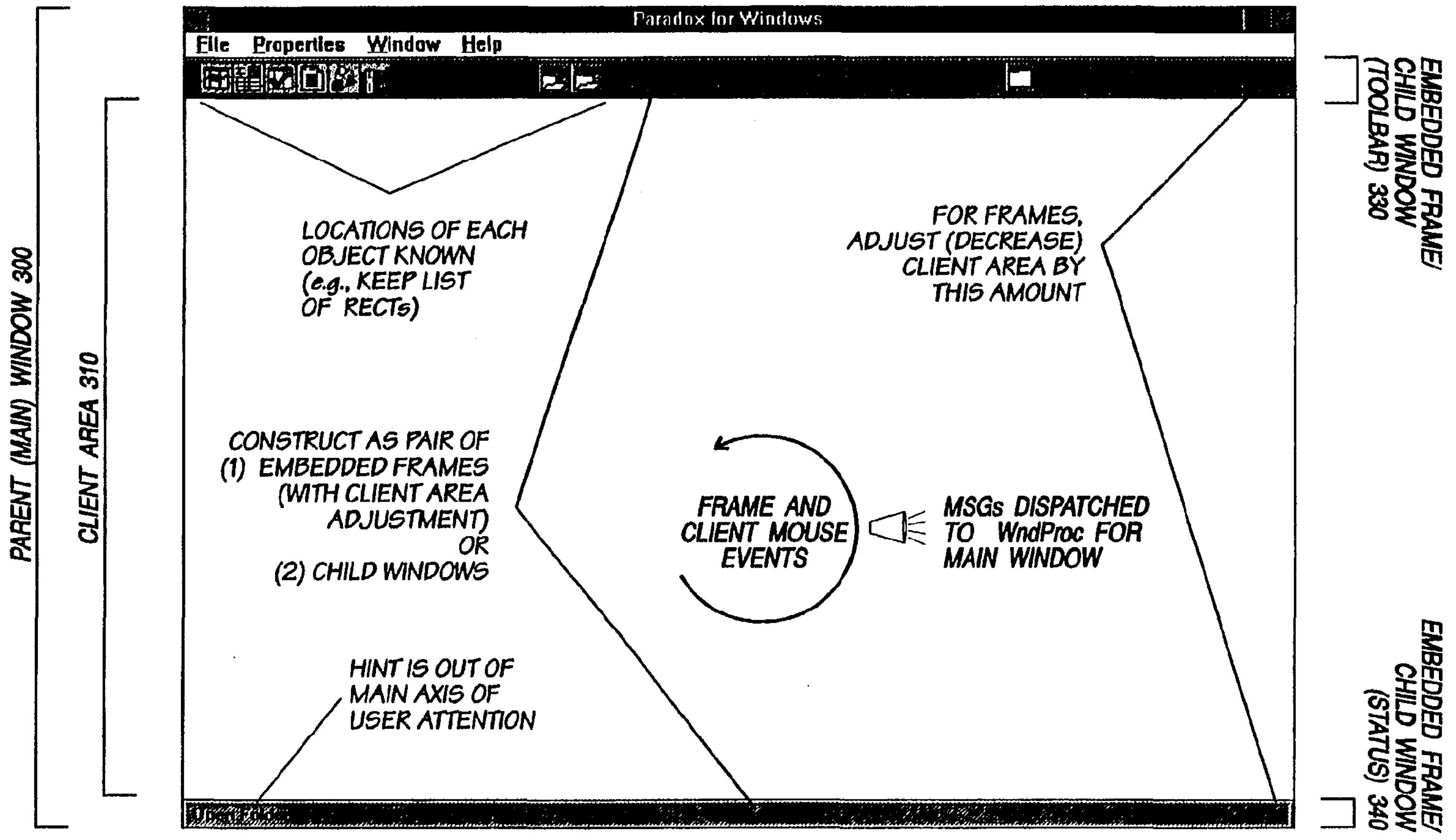

PARENT (MAIN) WINDOW 300
CLIENT AREA 310
Paradox for Windows
File Properties Window Help
EMBEDDED FRAME/ CHILD WINDOW (TOOLBAR) 330
LOCATIONS OF EACH OBJECT KNOWN (e.g., KEEP LIST OF RECTs)
FOR FRAMES, ADJUST (DECREASE) CLIENT AREA BY THIS AMOUNT
CONSTRUCT AS PAIR OF (1) EMBEDDED FRAMES (WITH CLIENT AREA ADJUSTMENT) OR (2) CHILD WINDOWS
FRAME AND CLIENT MOUSE EVENTS
MSGs DISPATCHED TO WndProc FOR MAIN WINDOW
HINT IS OUT OF MAIN AXIS OF USER ATTENTION
EMBEDDED FRAME/ CHILD WINDOW (STATUS) 340

FIG. 3A

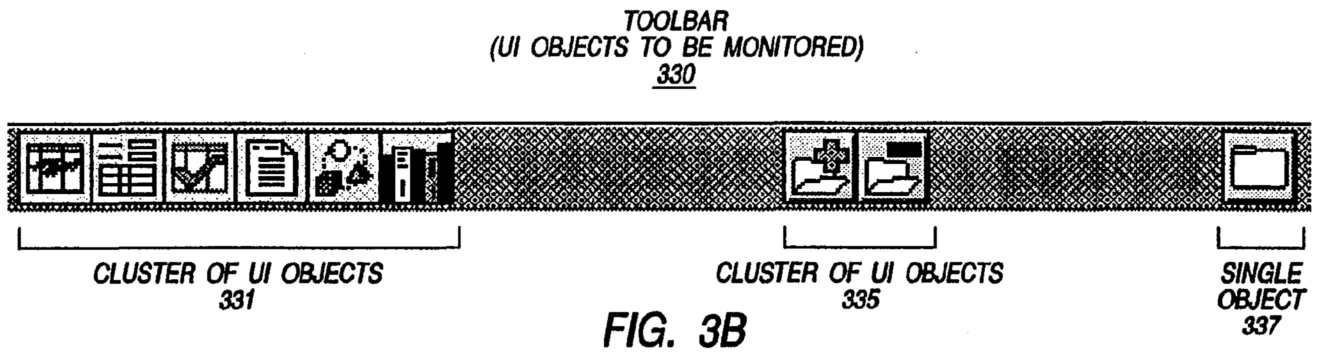


FIG. 3B

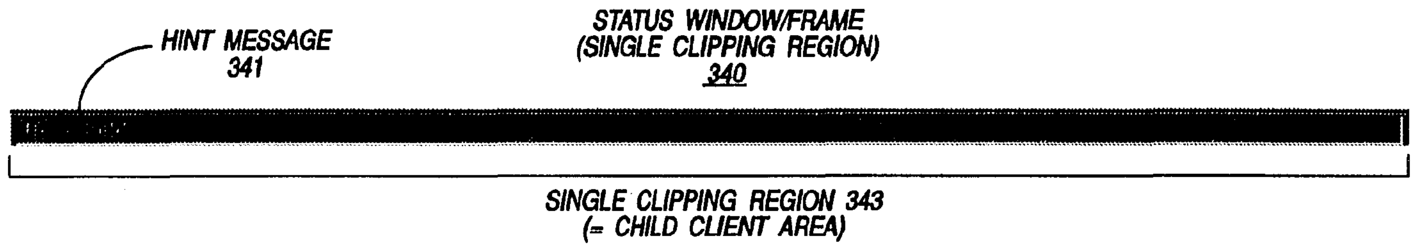


FIG. 3C

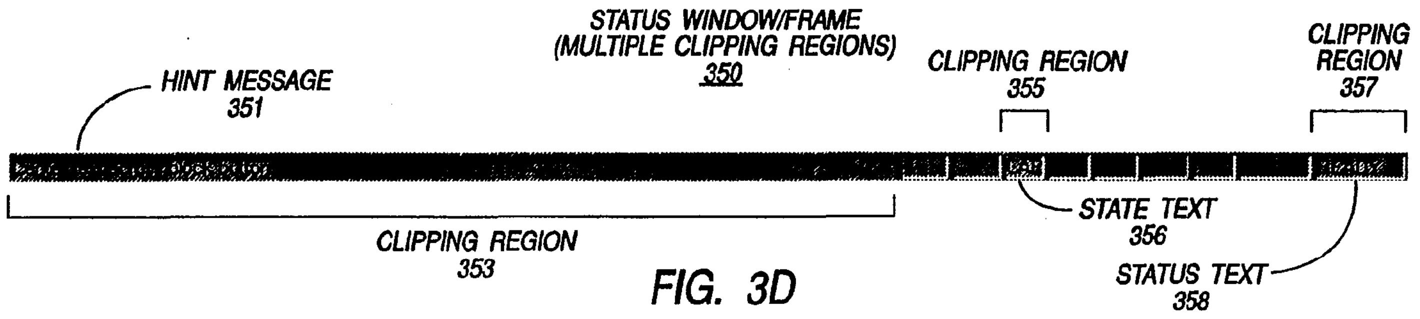
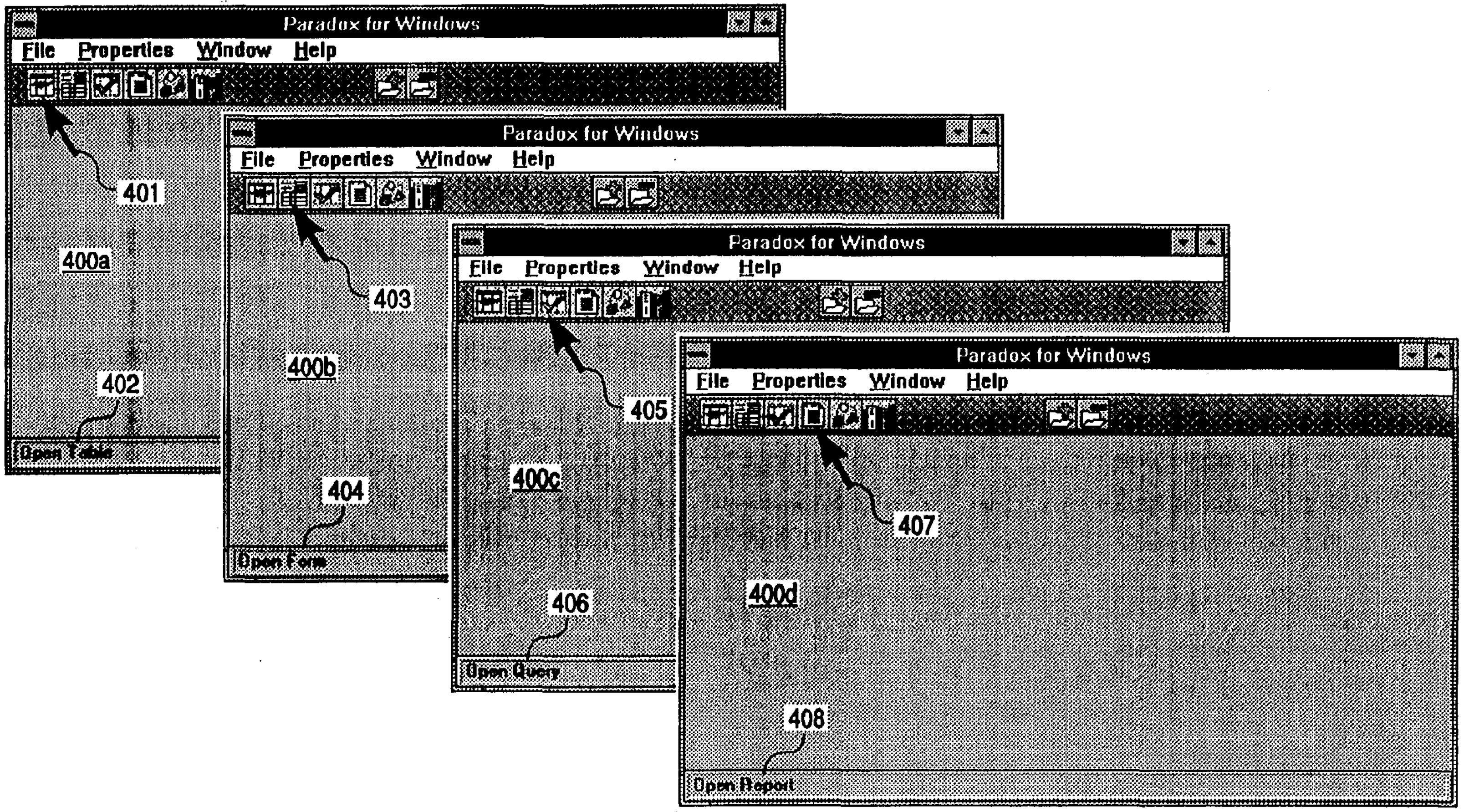


FIG. 3D



Paradox for Windows
File Properties Window Help
401
400a
402
Open Table
403
400b
404
Open Form
405
400c
406
Open Query
407
400d
408
Open Report

FIG. 4A

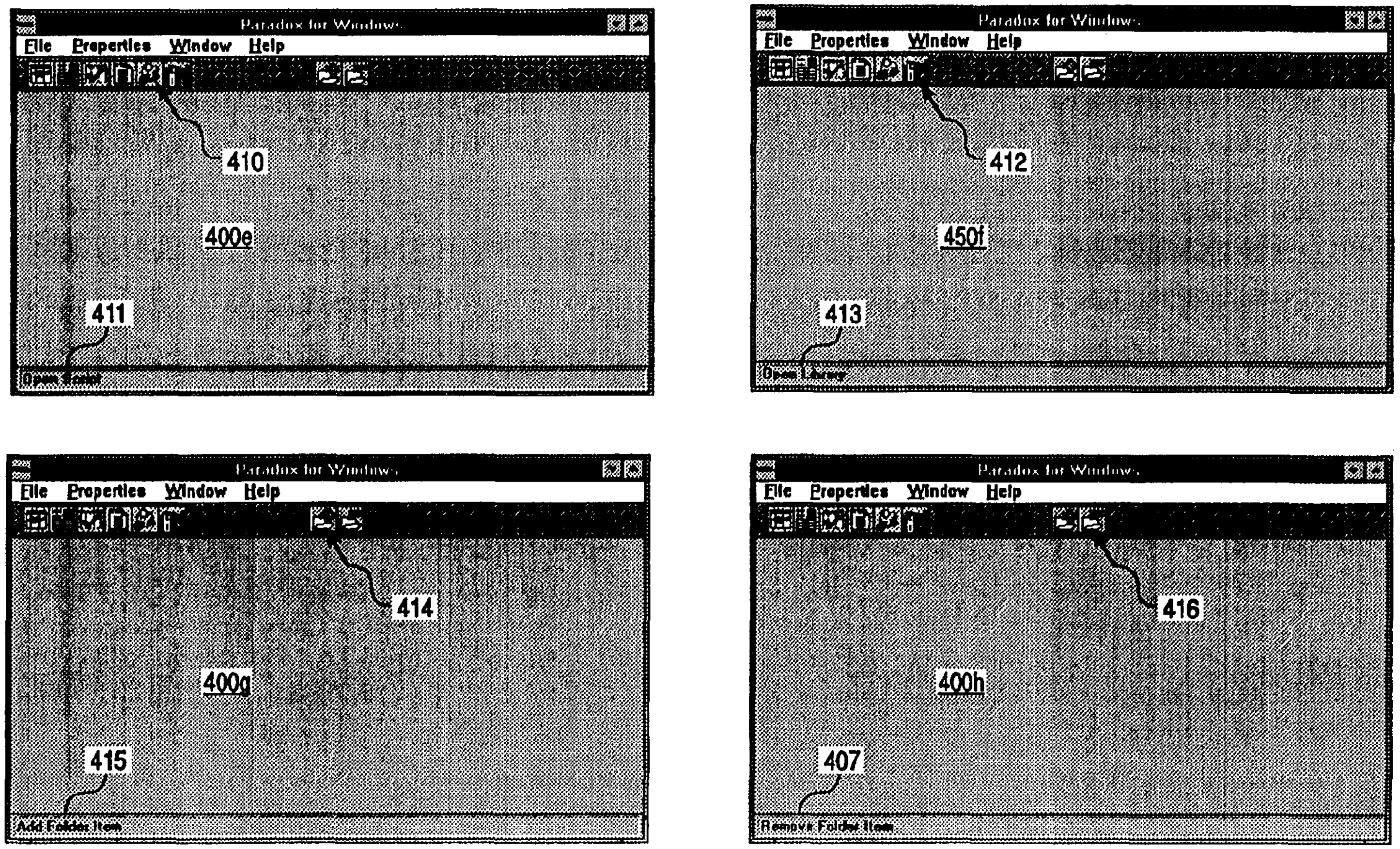

Paradox for Windows
File Properties Window Help
410
400e
411
Open Script
Paradox for Windows
File Properties Window Help
412
450f
413
Open Library
Paradox for Windows
File Properties Window Help
414
400g
415
Add Folder Item
Paradox for Windows
File Properties Window Help
416
400h
407
Remove Folder Item

FIG. 4B

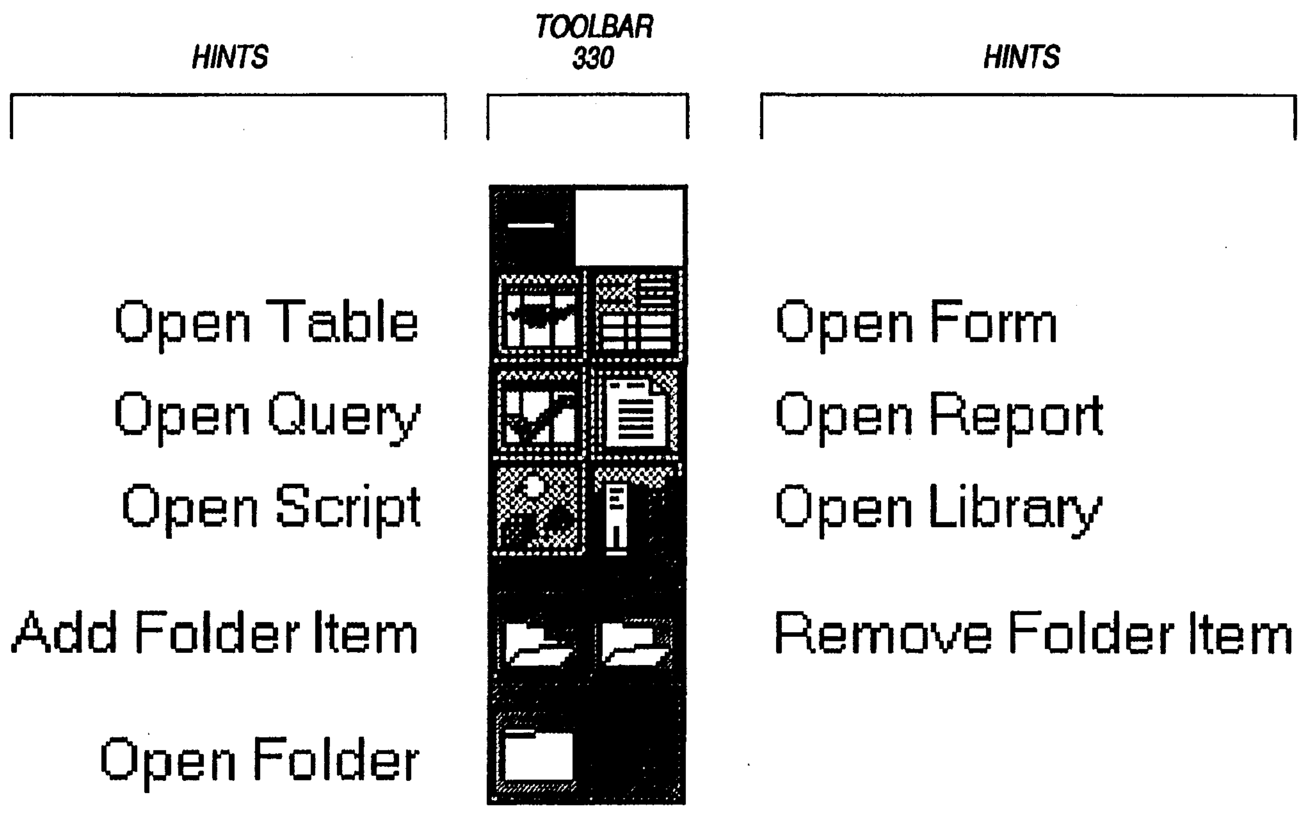
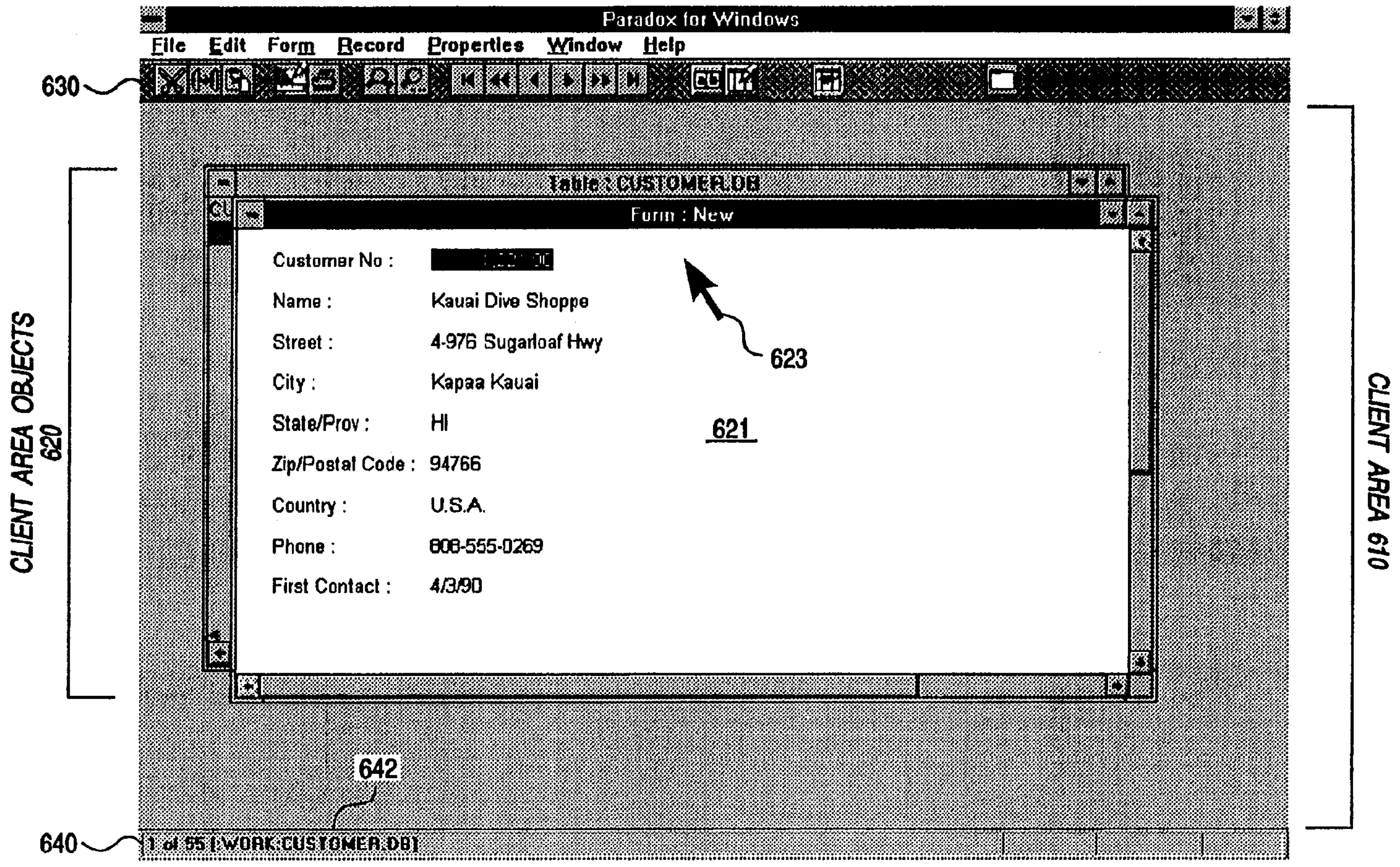
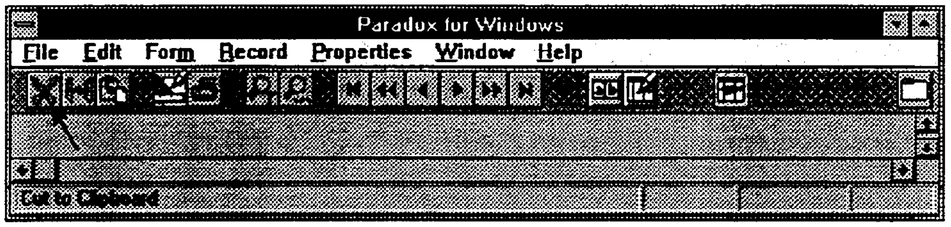

HINTS
TOOLBAR 330
HINTS
Open Table
Open Query
Open Script
Add Folder Item
Open Folder
Open Form
Open Report
Open Library
Remove Folder Item

FIG. 4C

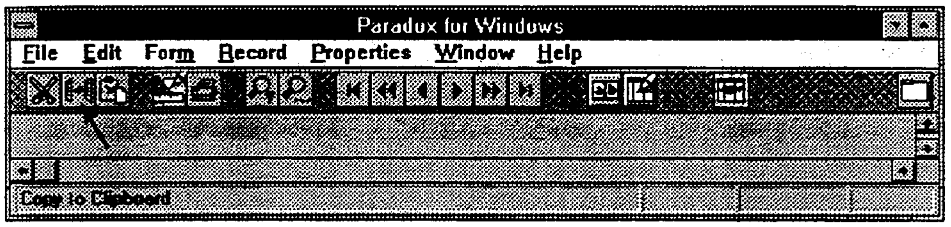


Customer No :	[illegible]
Name :	Kauai Dive Shoppe
Street :	4-976 Sugarloaf Hwy
City :	Kapaa Kauai
State/Prov :	HI
Zip/Postal Code :	94766
Country :	U.S.A.
Phone :	808-555-0269
First Contact :	4/3/90

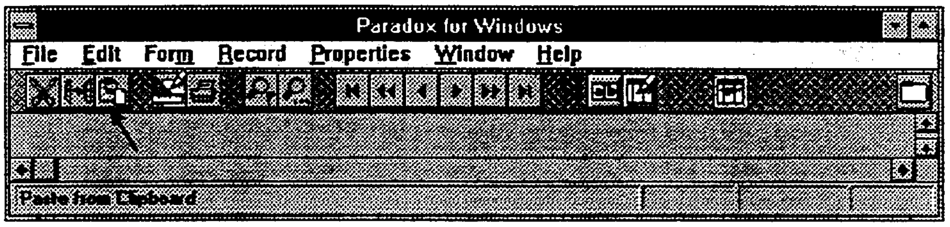
FIG. 5A


Paradox for Windows
File Edit Form Record Properties Window Help
Cut to Clipboard

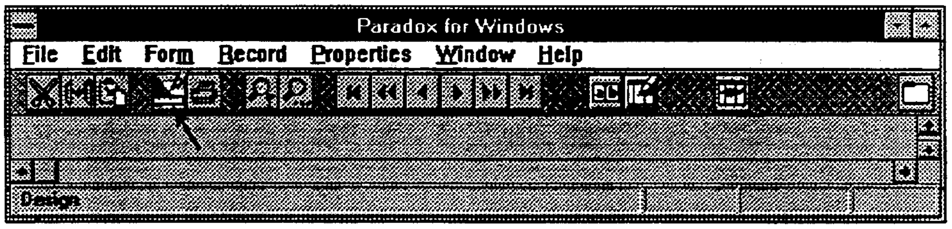
(a)


Paradox for Windows
File Edit Form Record Properties Window Help
Copy to Clipboard

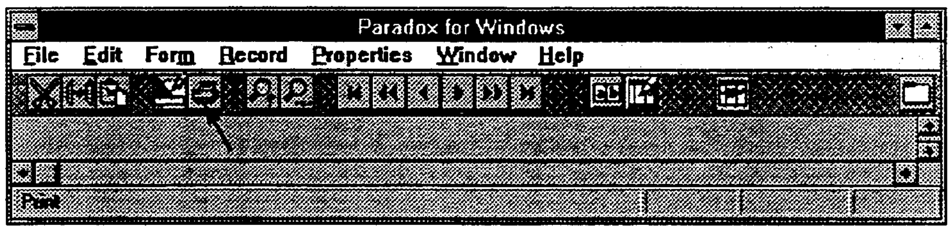
(b)


Paradox for Windows
File Edit Form Record Properties Window Help
Paste from Clipboard

(c)

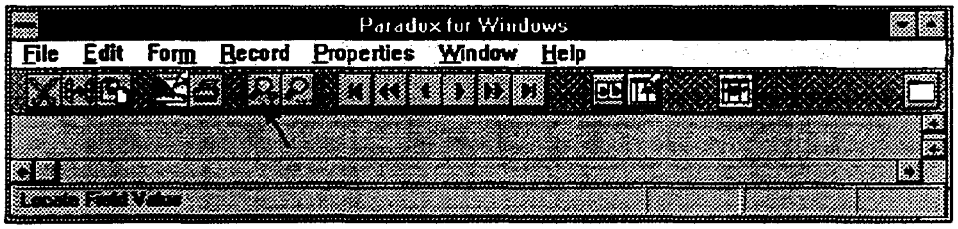

Paradox for Windows
File Edit Form Record Properties Window Help
Design

(d)

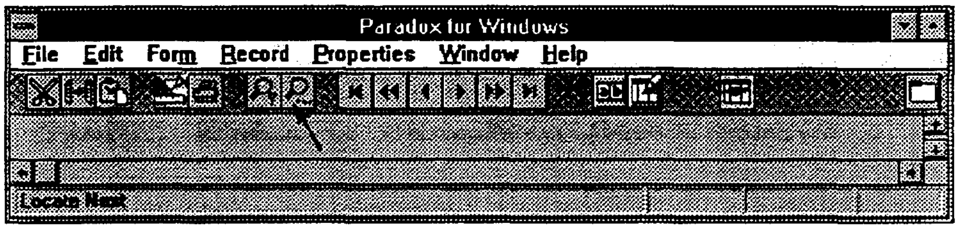

Paradox for Windows
File Edit Form Record Properties Window Help
Print

(e)

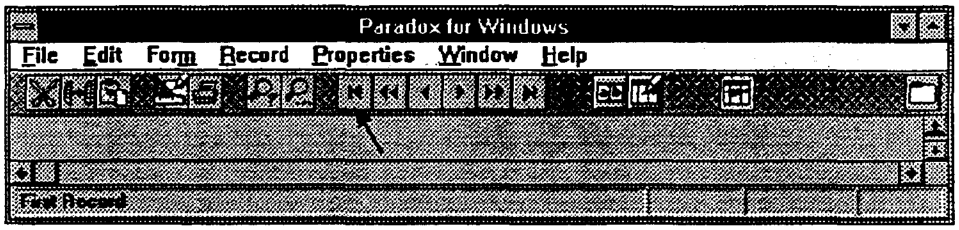
FIG. 5B



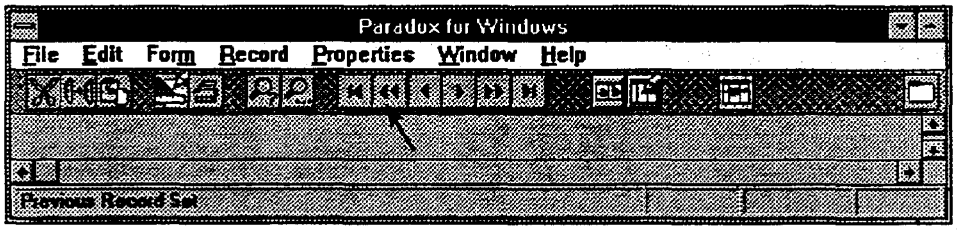
(f)



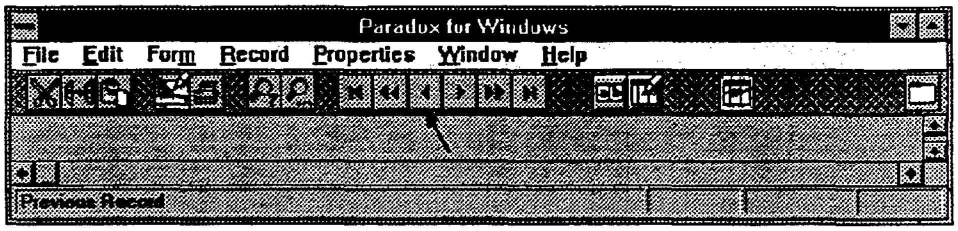
(g)



(h)

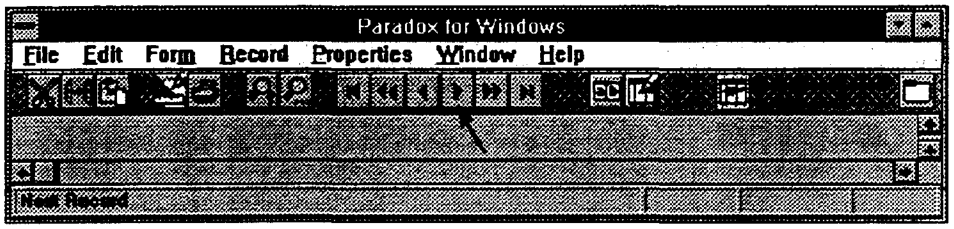


(i)

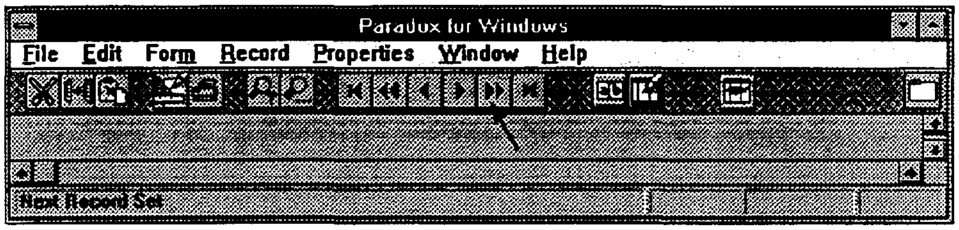


(j)

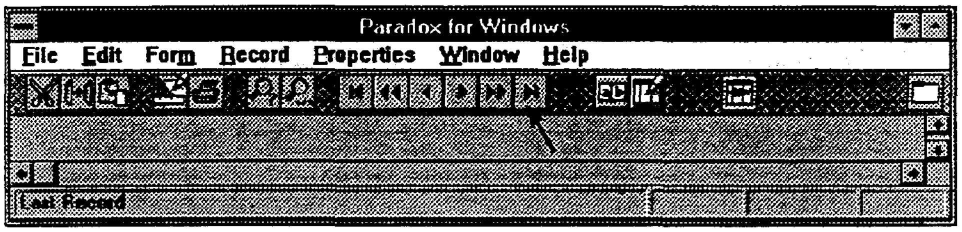
FIG. 5B (cont.)



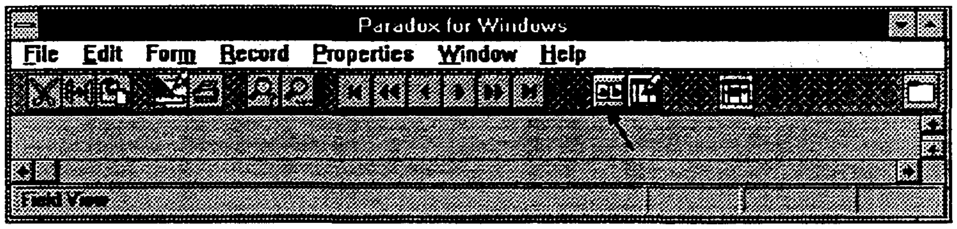
(k)



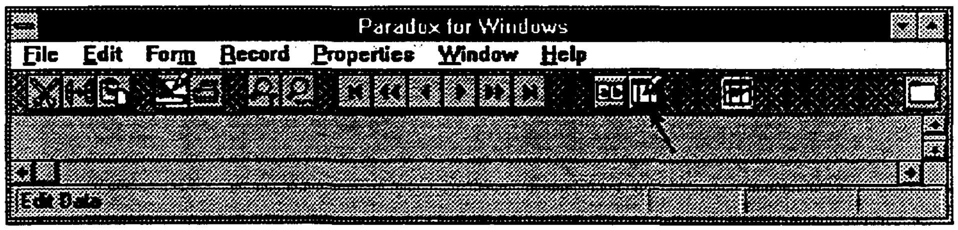
(l)



(m)

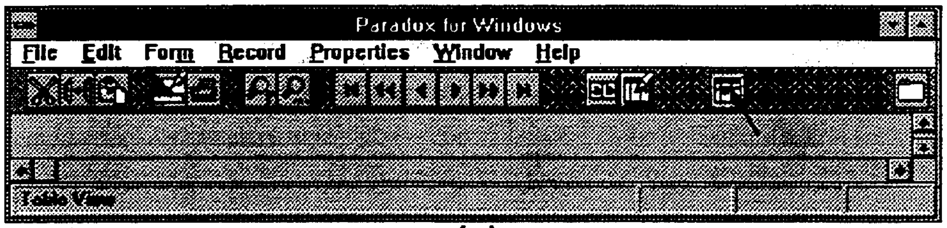


(n)



(o)

FIG. 5B (cont.)



(p)

FIG. 5B (cont.)

FORM DESIGN TOOLBAR
WITH HINTS

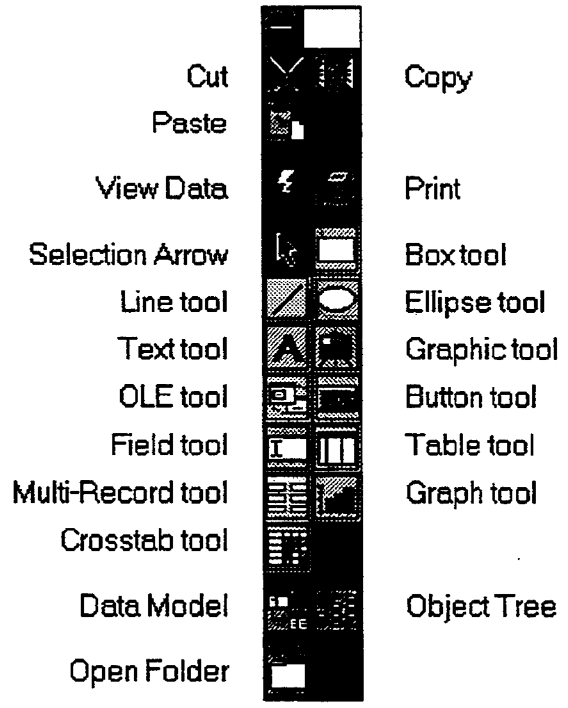


FIG. 5C

REPORT DESIGN TOOLBAR
WITH HINTS

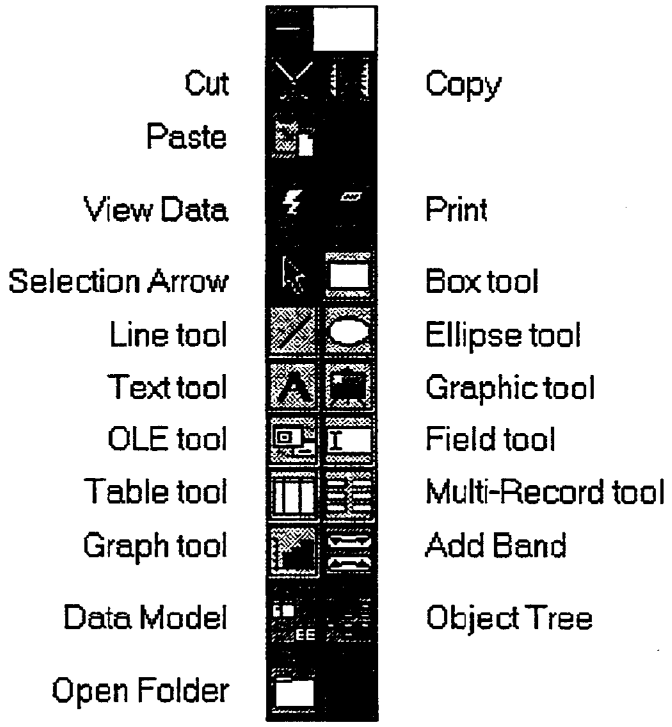


FIG. 6A

QUERY WINDOW TOOLBAR
WITH HINTS

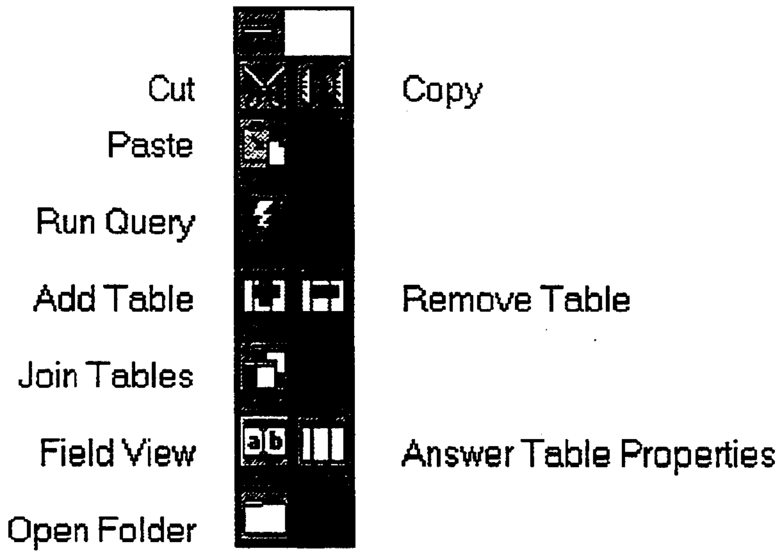


FIG. 6B

TABLE WINDOW TOOLBAR
WITH HINTS

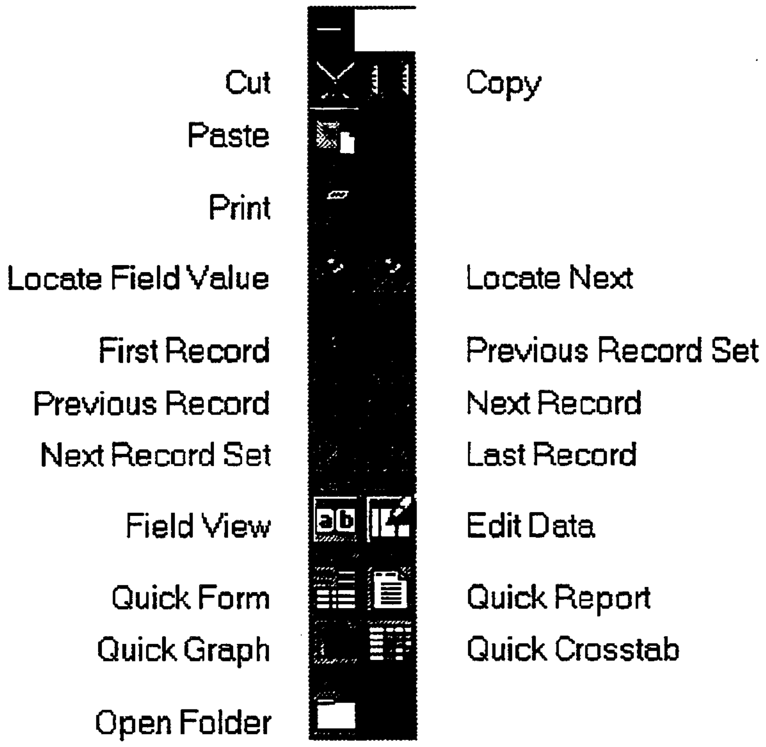


FIG. 6C

LIBRARY WINDOW TOOLBAR
WITH HINTS

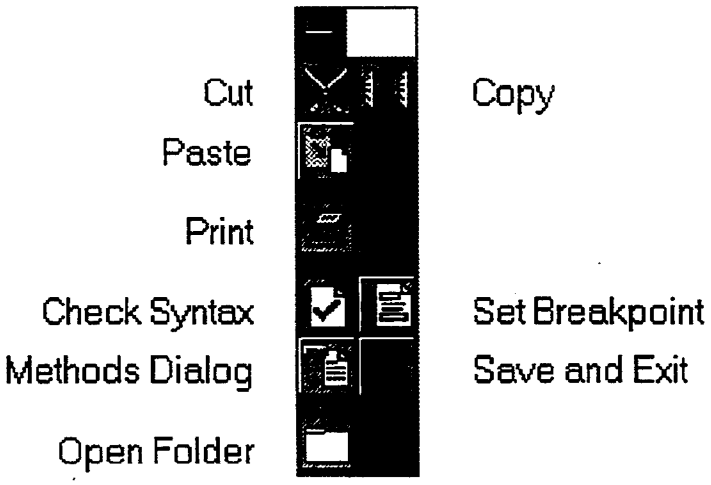


FIG. 6D

PROGRAMMING EDITOR TOOLBAR
WITH HINTS

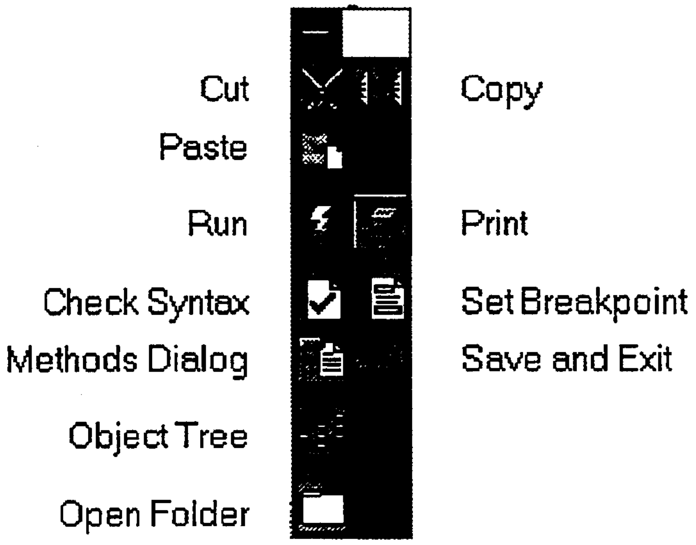


FIG. 6E

DEBUGGER TOOLBAR
WITH HINTS

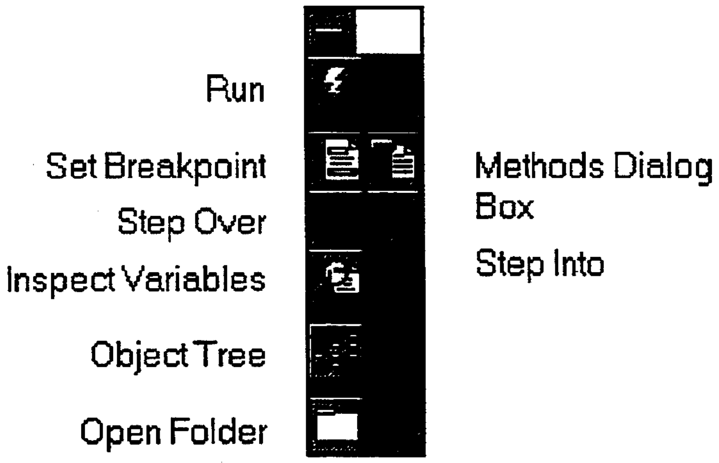


FIG. 6F

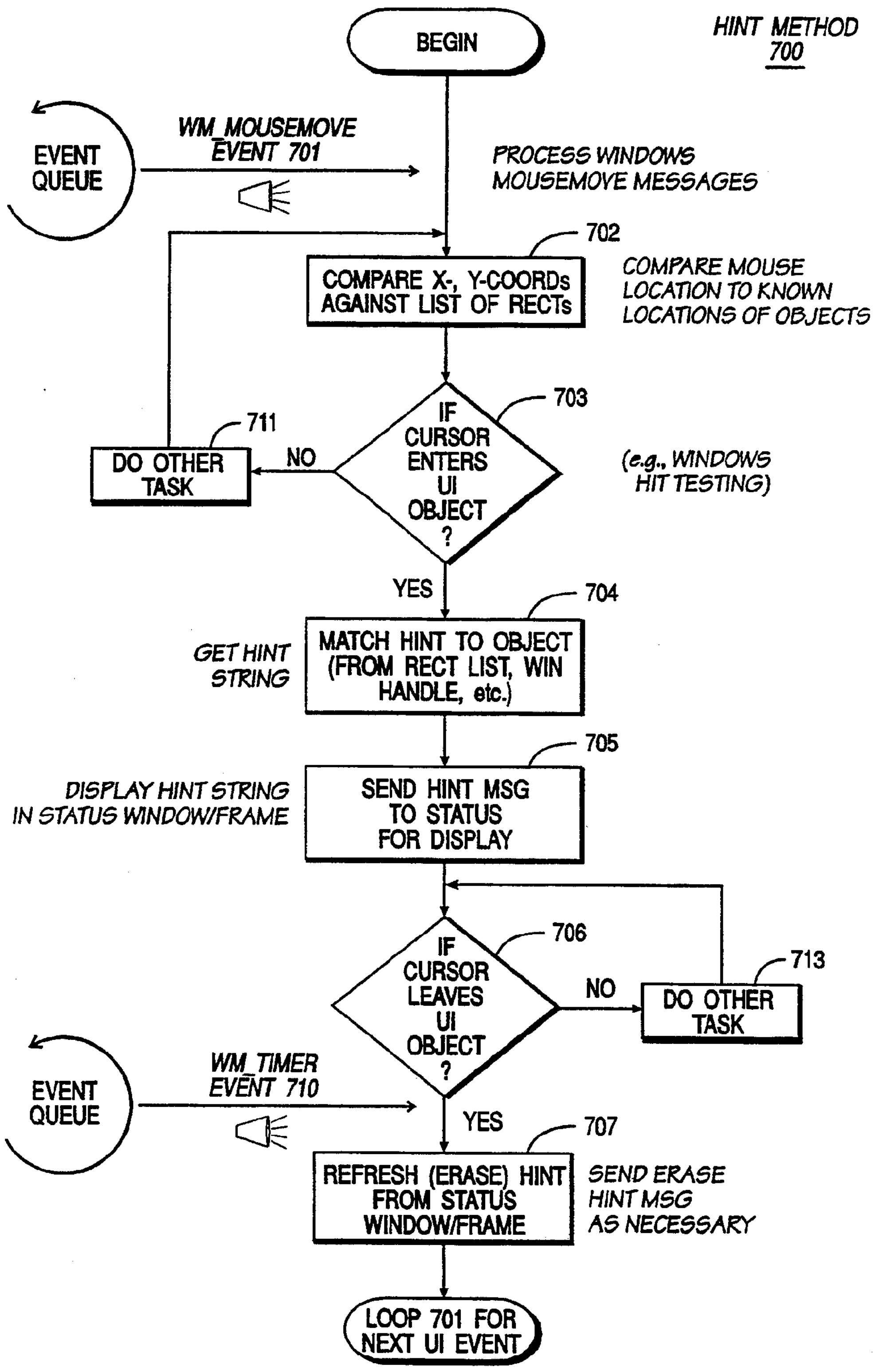

HINT METHOD 700
BEGIN
EVENT QUEUE
WM_MOUSEMOVE EVENT 701
PROCESS WINDOWS MOUSEMOVE MESSAGES
702
COMPARE X-, Y-COORDs AGAINST LIST OF RECTs
COMPARE MOUSE LOCATION TO KNOWN LOCATIONS OF OBJECTS
703
IF CURSOR ENTERS UI OBJECT ?
711
DO OTHER TASK
NO
(e.g., WINDOWS HIT TESTING)
YES
704
GET HINT STRING
MATCH HINT TO OBJECT (FROM RECT LIST, WIN HANDLE, etc.)
705
DISPLAY HINT STRING IN STATUS WINDOW/FRAME
SEND HINT MSG TO STATUS FOR DISPLAY
706
IF CURSOR LEAVES UI OBJECT ?
NO
713
DO OTHER TASK
EVENT QUEUE
WM_TIMER EVENT 710
YES
707
REFRESH (ERASE) HINT FROM STATUS WINDOW/FRAME
SEND ERASE HINT MSG AS NECESSARY
LOOP 701 FOR NEXT UI EVENT

FIG. 7

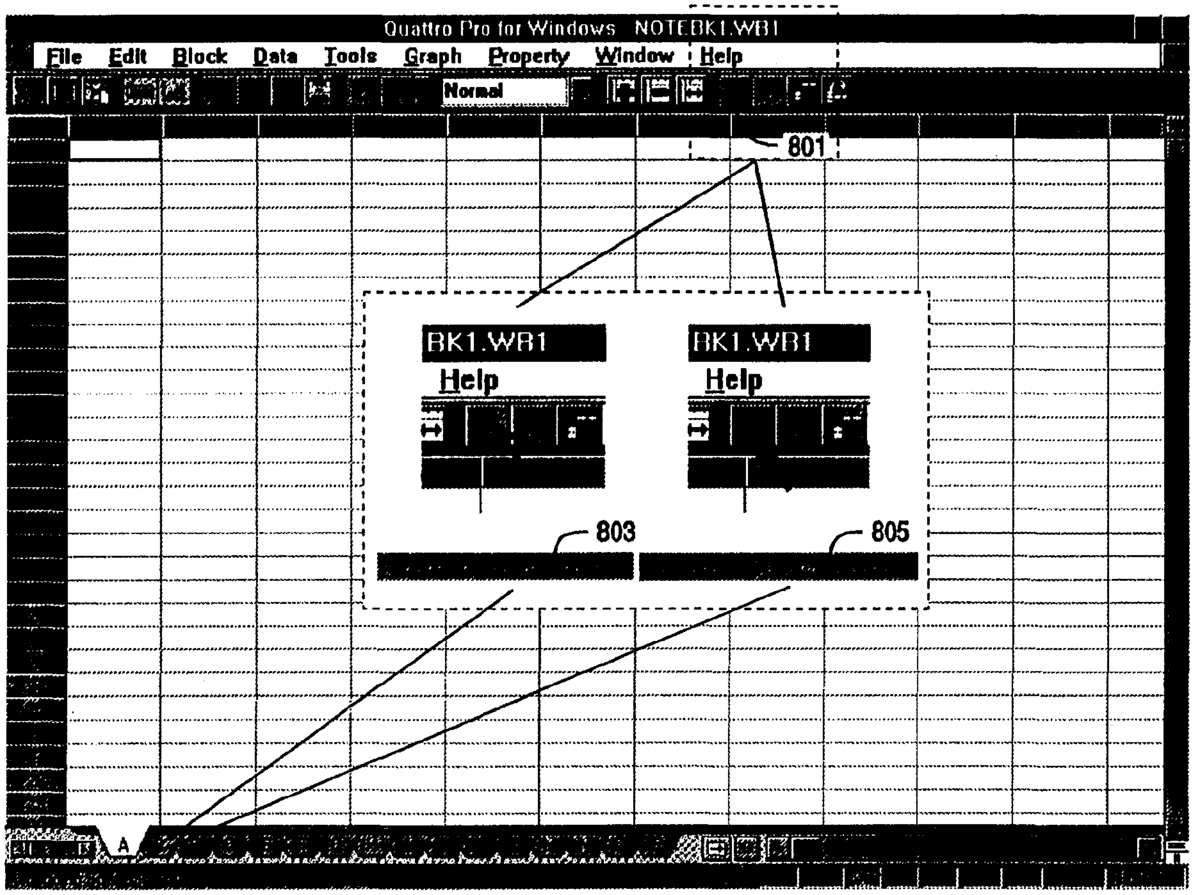

Quattro Pro for Windows NOTEBK1.WB1
File Edit Block Data Tools Graph Property Window Help
Normal
801
BK1.WB1
Help
BK1.WB1
Help
803
805
A

FIG. 8

GRAPHICAL USER INTERFACE SYSTEM AND METHODS FOR IMPROVED USER FEEDBACK

COPYRIGHT NOTICE

BACKGROUND OF THE INVENTION

The present invention relates generally to computer systems and, more particularly, to systems and techniques for assisting a user of a computer system.

With the advent of the personal computer, the use of computer systems is becoming increasingly prevalent in everyday life. In the past, computers were often housed in highly restricted areas, with access limited to a few computer scientists and programmers. Today, however, computers can be seen on the desktops of most business professionals. Running software applications such as word processors and spreadsheets, for example, even the average business professional can realize substantial productivity gains. Besides the business environment, computers can also be found in wide use both at home and at school.

The typical user of a computer today has little or no training in the computer sciences or even in the basic use of a personal computer. In order to operate a computer effectively, however, he or she must overcome a steep learning curve, one requiring mastery of a number of commands and data formats. One approach to this problem is to spend hours laboring over often-cryptic user manuals—an unattractive option at best. Instead, most users usually abandon printed manuals in favor of trial-and-error learning.

To increase ease of use, designers of computer systems have labored for decades to create architectures which are intuitive. Most of this effort has been centered around the user interface or UI—the means by which a user communicates (i.e., supplies input and receives output) with a computer. With increasingly widespread availability of powerful microprocessors, graphical user interfaces (GUIs, pronounced "gooeys") have become feasible.

A GUI is a type of display format that enables a user to operate a computer by pointing to pictorial representations, such as "windows" and "icons" (bitmaps), displayed on a screen device. A window is a rectangle displayed on the screen that affords a user workspace within a program. In typical operation, the user may move the window about on the screen, change its size or shape, enlarge it to fill the screen, close it entirely, or change how much of its contents are displayed. To aid the user in the manipulation of its contents, a window will typically include a number of user interface components, such as buttons, menus, sliders, and the like. Outside the window, the screen can display other screen objects, such as other windows, disk drive icons, or even a trash can icon.

To navigate within a GUI, most systems employ a screen cursor or pointer, typically displayed as a small arrow icon (bitmap) which allows the user to select individual points on the screen. In operation, the screen cursor is moved to a desired screen location in response

to movements of a pointing device (e.g., mouse) by the user. Besides effecting cursor movement, most pointing devices include one or more switches or "mouse buttons" for specifying additional user input or "user events." Since many user choices may be entered through use of a pointing device (e.g., for selecting screen objects), instead of input with a keyboard, the need for the user to memorize special commands has been lessened.

Most GUIs feature a menu bar, for instance, running across the top of the screen which serves to group or categorize commands available to the user. Clicking on an item on the menu bar typically causes a "pull-down" menu to appear. This second or "submenu" also includes a number of items, each of which is associated with a desired action, including the display of even more menus. To select a desired action, the user usually clicks the corresponding menu item with the screen or mouse pointer. For some menu items, particularly those which may be nested in several layers deep, a keyboard equivalent or "hot key" may be available but, unfortunately, these must also be memorized by the user.

With well-known examples including Apple's Macintosh (Mac) interface, Microsoft's Windows, and IBM's OS/2 Presentation Manager, these interfaces simplify computer operation by providing users with a more-or-less consistent interface across applications. Nevertheless, most application software still requires complex user actions, such as "double-clicking" or "dragging" a mouse device while a key is held down. Thus, there typically exists a plethora of ways to do almost anything in a graphical interface, such as the Mac. While this increases the flexibility of a system, it also adds to the complexity of the interface that the user must master. And this problem is by no means limited just to novice users. Experienced computer users are reluctant to read user manuals and, hence, often fair no better than novice users. All told, the user is still required to memorize special commands.

Standard windowing interfaces depend heavily on a clunky system of pull-down menus. While pull-down menus are an improvement over command-line (e.g., MS-DOS) interfaces, they are non-metaphoric or non-analogous to ordinary objects, i.e., ones familiar to the user. Perhaps the biggest weakness of pull-down menus is the requirement that the user must know beforehand which menu contains the item or function of interest. If the user does not know which pull-down menu, submenu, or dialog box contains the item he or she is seeking, the user will spend an inordinate amount of time checking the contents of each in an effort to hunt down the needed item. And often the search is in vain. Moreover, since functions for a given object (e.g., text object) are often scattered over several disparate menus, the user is discouraged from interacting and experimenting with the object. Thus, many prior art windowing GUIs are no more intuitive or useful to the user than other menuing systems.

One approach to overcoming this problem has been the implementation of "smart icons." This interface technique employs screen buttons painted with icons which are supposed to tell the user what the buttons do. This approach, however, makes the interface problem even worse because the meaning of the individual buttons is often not easily grasped. For instance, some button images are so small or so numerous that it is hard to see the icons well enough to discern what they mean.

Thus, the approach is frequently no more helpful than hiding the desired function deep inside a menuing system. Thus, with advances in computer and software technology, application software has not necessarily become easier to use. Instead, technological advances have been largely employed to build more complex functions into existing applications, often resulting in a complicated user interface, such as a staggering array of icons, which leave the user even more bewildered.

The present invention recognizes that it is highly desirable to provide computers with system and application software which is highly intuitive to users, including those who are untrained in the use of the software. What is needed is a system and interface methods which require little or no knowledge of specific commands by the user. Moreover, such methods should not "invade" the user screen workspace (client area), that is, they should operate in a non-intrusive fashion. The present invention fulfills this and other needs.

SUMMARY OF THE INVENTION

The present invention recognizes that present-day implementations of graphical user interfaces often serve to further confuse a user of a computer system. Examples include toolbars or "smart" icons, which, owing to the vast number of bitmap images typically employed, force the user to memorize a multitude of (often arbitrary) images.

The present invention further recognizes that conventional techniques for providing screen feedback, such as "balloon help" and "intelligent cursors," operate in a fashion which is not only intrusive to the active workspace (client area), but is also computationally very inefficient. Balloon help, for instance, by "popping up" at a screen location proximate the cursor location, typically obscures the work product of interest. Moreover, the task of continually creating and destroying user interface windows (the "balloons"), being resource intensive, substantially degrades the performance of such systems. By overwriting the client area (i.e., active screen portion where the user's data is represented) and thereby destroying the screen image of the underlying work product, such systems must undertake the computationally expensive operation of "repainting" the user image.

According to the present invention, a graphical user interface includes improved interface components and methods for supplying the user with "hints" which enable users to completely discern the functionality of smart icons, interface objects, or other screen components of interest The "hints", which may be in the form of textual (or graphic) messages, are restricted to a non-intrusive location of the screen, typically located along one side (e.g., bottom or inferior portion) of a program's client area. When the screen cursor touches an object (e.g., enters a button) of interest, the system identifies the object with an appropriate descriptor hint displayed in the non-intrusive region.

In an exemplary method of the present invention, the entrance of the cursor into and departure from screen objects is continually monitored as follows. Upon entrance of the cursor into the domain of an object (e.g., boundary of a known screen region), the object is identified (e.g., from hit-testing, its window handle, or the like). Upon identification, a corresponding descriptor or "hint" may be determined. This hint is displayed in a status frame or window—a static frame (i.e., it is continually displayed) along a single portion (e.g., the bottom) of the client area. Upon departure of the cursor from the object, the descriptor is cleared or erased from the status window. In this manner, the system may continually monitor and describe system objects of interest in a non-intrusive and computationally frugal fashion.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1A** is a block diagram of a computer system in which the present invention may be embodied.

FIG. **1B** is a block diagram of a software system of the present invention, which includes operating system, application software, and user interface components.

FIG. **1C** is a bitmap screenshot illustrating the basic architecture and functionality of a graphical user interface in which the present invention may be embodied.

FIG. **2** is a pair of flowcharts contrasting the operation of conventional modal architecture with that of event-driven architecture.

FIG. **3A** is a bitmap screenshot illustrating an improved graphical user interface of the present invention.

FIGS. **3B–D** are bitmap illustrations further describing interface components from the graphical user interface of FIG. **3A**.

FIGS. **4A–B** are a series of bitmap screenshots which illustrate (in animation fashion) a "hint" interface method of the present invention.

FIG. **4C** is a bitmap which summarizes the hints for the toolbar interface component from the improved graphical user interface of the present invention.

FIG. **5A** is a bitmap screenshot which illustrates an exemplary operation of the graphical interface of the present invention, the example including client area objects which interact with a screen cursor.

FIG. **5B** is a sequence of bitmap screenshot frames illustrating hint methodology for the interface of FIG. **5A**.

FIG. **5C** is a bitmap summarizing the toolbar of FIG. **5A** with exemplary hints.

FIGS. **6A–F** are a series of bitmaps illustrating a multitude of user interface toolbars, each one employed for a particular task; the bitmaps include exemplary hints which may be employed with each toolbar.

FIG. **7** is a flowchart illustrating a hint method of the present invention.

FIG. **8** is a bitmap screenshot illustrating multiple hints for a screen object having multiple regions.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

The following description will focus on the presently preferred embodiment of the present invention, which is operative in the Microsoft® Windows environment. The present invention, however, is not limited to any particular one application or any particular windows environment. Instead, those skilled in the art will find that the system and methods of the present invention may be advantageously applied to a variety of system and application software, including database management systems, wordprocessors, spreadsheets, and the like. Moreover, the present invention may be embodied on a variety of different platforms, including Macintosh, UNIX, NextStep, and the like. Therefore, the description of the exemplary embodiments which follows is for purposes of illustration and not limitation.

System Hardware

As shown in FIG. **1A**, the present invention may be embodied in a computer system such as the system **100**, which comprises a central processor **101**, a main memory **102**, an input/output controller **103**, a keyboard **104**, a pointing device **105** (e.g., mouse, track ball, pen device, or the like), a display device **106**, and a mass storage **107** (e.g., hard disk). Additional input/output devices, such as a printing device **108**, may be included in the system **100** as desired. As illustrated, the various components of the system **100** communicate through a system bus **110** or similar architecture. In a preferred embodiment, the computer system **100** includes an IBM-compatible personal computer, which is available from several vendors (including IBM of Armonk, N.Y.).

System Software

A. Overview

Illustrated in FIG. 1B, a computer software system **120** is provided for directing the operation of the computer system **100**. Software system **120**, which is stored in system memory **102** and on disk memory **107**, includes a kernel or operating system (OS) **140** and a windows shell **150**. One or more application programs, such as application software **125** or windows application software **145**, may be "loaded" (i.e., transferred from storage **107** into memory **102**) for execution by the system **100**.

System **120** includes a user interface (UI) **160**, preferably a graphical user interface (GUI), for receiving user commands and data. These inputs, in turn, may be acted upon by the system **100** in accordance with instructions from operating module **140**, windows **150**, and/or application modules **125**, **145**. The UI **160** also serves to display the results of operation, whereupon the user may supply additional inputs or terminate the session. Although shown conceptually as a separate module, the UI is typically provided by windows shell **150**, operating under OS **140**. In a preferred embodiment, OS **140** is MS-DOS and windows **150** is Microsoft ® Windows; both are available from Microsoft Corporation of Redmond, Wash.

System **120** also includes a hints module **130** of the present invention for aiding users of the computer **100**. As shown, the module **130** is typically constructed within applications **145**. Before undertaking a detailed description of the construction and operation of the hints module **130** itself, however, it is helpful to first examine the general methodology of UI **160** and the event-driven architecture driving its operation.

B. Graphical User (Windowing) Interface

As shown in FIG. 1C, the system **100** typically presents UI **160** as a windowing interface or workspace **161**. Window **161** is a rectangular, graphical user interface (GUI) for display on screen **106**; additional windowing elements may be displayed in various sizes and formats (e.g., tiled or cascaded), as desired. At the top of window **161** is a menu bar **170** with a plurality of user-command choices, each of which may invoke additional submenus and software tools for use with application objects. Window **161** includes a client area **180** for displaying and manipulating screen objects, such as graphic object **181** and text object **182**. In essence, the client area is a workspace or viewport for the user to interact with data objects which reside within the computer system **100**.

Windowing interface **161** includes a screen cursor or pointer **185** for selecting and otherwise invoking screen objects of interest. In response to user movement signals from the pointing device **105**, the cursor **185** floats (i.e.,

freely moves) across the screen **106** to a desired screen location. During or after cursor movement, the user may generate user-event signals (e.g., mouse button "clicks" and "drags") for selecting and manipulating objects, as is known in the art. For example, Window **161** may be closed, resized, or scrolled by "clicking on" (selecting) screen components **172**, **174/5**, and **177/8**, respectively. Keystroke equivalents, including keyboard accelerators or "hot keys", are provided for performing these and other user operations through keyboard **104**.

C. Event-driven Architecture

Underlying the windowing interface is a message or event-driven architecture. This model is perhaps best described by contrasting its operation with that of a modal or sequential architecture which has been traditionally employed. In this manner, the reader may appreciate the added flexibility of a message-driven system—flexibility which is employed by the system of the present invention for providing object-specific hints for objects which the screen cursor touches.

As shown in FIG. 2, a modal program **200** comprises a series of discrete operating blocks or modes **210**, with a well-defined beginning, middle and end. In actual operation, such a program typically displays a series of input screens for receiving user information, for example, to create a written report. For instance, the first entry screen may require a customer name, the second a customer address, the third a part number, and so on. The program typically terminates in an output mode, for reporting results determined from the various inputs. Thus, the program **200** follows a fairly rigid sequence of operation, with each input or entry mode demanding successful completion before the program proceeds to the next step.

While a modal program is relatively easy to design and implement, it is not so easy to use. The design certainly ensures that all required information is entered, but only at the expense of forcing users to operation in a manner dictated by the program. Specifically, since the program is built around a pre-arranged set of modes, a user cannot get from one mode to another without first completing a previously-required mode. In the present example, for instance, a user must needlessly complete a customer name entry screen (and any other intervening input screens) just to access part number information. Any deviation from this sequence by the user is simply not permitted. Lacking flexibility, modal programs make a poor choice for handling real-world tasks.

As shown in the second half of FIG. 2, an event-driven architecture **250** eschews a pre-selected sequence, opting instead for an "event loop." The event loop **260** is a centralized mechanism for processing messages about user and system events. It includes an event queue **270** and mechanisms for retrieving **263** and dispatching **269** messages to various window classes **280**. Before each of these components is described in detail, it is helpful to have a firm understanding of "messages."

In a typical modal environment, especially those typified by a character-based UI, a program reads from the keyboard by making an explicit call to a function, such as the C function getchar(). The function typically waits until the user presses a key before returning the character code to the program; all system activity ceases until completion of this one step. In a Windows environment, in contrast, the operating system uses messages to manage and synchronize multiple applications and hardware events, such as clicks of a mouse or presses of a keyboard, which are converted to messages by Windows event handlers.

From a programming perspective, a message is simply a data structure containing information about a particular event. In Microsoft Windows, a message is a 16-bit unsigned integer which serves as a symbolic constant for a particular event; packaged within this integer is a message identifier and message parameters, which vary with each event type represented. For example, messages from a window object might include information about creating (WM_CREATE), closing (WM_CLOSE), moving (WM_MOVE), and re-sizing (WM_SIZE) the window. The input messages are collected in a system-wide queue and then directed to the proper window. These messages, along with timer and screen paint messages, must be passed to the target application(s) of interest.

A mechanism is provided for retrieving messages from the system queue and dispatching them to the appropriate application which, in turn, may proceed to process any message that arrives. Each window belongs to a certain window type or "class" which defines certain characteristics common to all windows of that type. Associated with each type is a Windows function which processes all messages sent to windows of its type. An application queue is provided where Windows may place messages that belong to a specific application. When the application is ready to receive input, it simply reads the awaiting messages. If none are found or if there exists a message for other applications with higher priority, Windows passes control to the other applications.

The message or event loop **260** shown in FIG. **2**, for example, may be simply coded as:

```
while (GetMessage (&msg, NULL, 0, 0))
        {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
        }
return msg.wParam ;
}
```

where a message (&msg) is retrieved by a call to GetMessage (step **263**); if needed, the retrieved message may be translated by a call to TranslateMessage() and then dispatched by a call to DispatchMessage (step **269**). This "while" loop continues until the GetMessage function returns a value of zero—indicating that the loop has read a WM_QUIT message from the queue, telling the application to end (yes at step **265**).

In addition to the messages generated by Windows, applications can create their own messages and place them in the application queues of other applications. The SendMessage and PostMessage functions let applications pass messages to their windows or to the windows of other applications. The PostMessage function directs Windows to post the message by placing it in the application queue. Control returns immediately to the calling application, and any action to be carried out as a result of the message does not occur until the message is read from the queue.

The SendMessage function, on the other hand, directs Windows to send a message directly to the given Windows function, thereby bypassing the application queue. Windows does not return control to the calling application until the Windows function that receives the

message has processed it. Such an unqueued message, however, is generally a message that affects the window only.

The general mechanism for retrieving and dispatching messages in an event-based system, such as Microsoft Windows, is known in the art; see, e.g., Petzold, C., *Programming Windows*, Second Edition, Microsoft Press, 1990. Additional information can be found in Microsoft's Window Software Development Kit, including: 1) *Guide to Programming*, 2) *Reference*, Vols. 1 and 2, and 3) *Tools*, all available from Microsoft Corp. of Redmond, Wash. The disclosures of each of the foregoing are hereby incorporated by reference.

Preferred Graphical User Interface

Referring now to FIG. 3A, a preferred graphical user interface of the present invention will be described. As shown, the interface includes a parent or main window **300** having a client area **310**, in a fashion similar to that for the above-described Windows interface **161** (in FIG. 1C). In addition, however, the main window **300** includes a plurality of child windows or "toolbar" **330** and a "status" child window **340**. Each will be examined in turn.

With specific reference to FIG. 3B, the toolbar **330** comprises a plurality of "custom control" buttons, typically positioned just below the main menu. Providing quick access to the equivalents of menu commands (i.e., word on a menu one chooses to perform an action), the toolbar greatly speeds up a given task, such as navigating through one's data. As shown, selected ones of the buttons may be clustered together (e.g., cluster **331** and cluster **335**) according to (similar) functionality. Alternatively, buttons may be separated (e.g., control or button **337**). Thus, a toolbar typically appears as set of bitmaps or icons below the menu line in an application's main window. It may contain buttons, edit fields, combo boxes, and the like. In a well-designed program, controls are grouped by related functionality.

The client area may be divided into several smaller logical areas. By using "hit-testing"—determining where on the screen the user is pointing at with the cursor—a program may determine the current location of the cursor. In general, hit-testing involves calculations using the X- and Y- coordinates passed to a Windows window procedure (in the 1Param parameter of the mouse message). Hit-testing is simplified through the use of "child windows." The child windows divide the entire client area into several smaller rectangular regions. Each child window has its own window handle, window procedure, and client area. Each window procedure receives mouse messages that apply only to its child window.

A particular application of child windows is for creating "child window controls." From a programmer's viewpoint, toolbars are child windows located at the top of a main window's client area. They extend across the width of the client area and are automatically re-sized whenever the main window changes shape. The program's other windows inhabit the remaining client area of the main window, typically below the toolbar. A wordprocessing application, for instance, will have one or more document windows open in the client area. These additional windows are typically displayed in the portion of the client area which is not occupied by the toolbar.

In the Windows programming environment, therefore, the most basic building block of the user interface construction is the "control." In Windows, a control is a standard child window that has been registered to participate as a component of the graphical user interface. Controls may receive input, display output, or perform both input and output. Windows provides built-in support for six common control classes: button, edit, list box, combo box, scroll bar, and static. Controls are characterized by their high degree of specialization and focus. The button control class, for instance, simply presents a clickable box for yes or no and multiple-choice input. To this, one can add his or her own "custom controls" to augment these common classes.

Toolbars are typically constructed from "owner-drawn" buttons. These differ from standard buttons only in that the application (not the operating system) assumes full responsibility for the button, including painting it, setting input focus, selecting it, and enabling or disabling it. Thus, a custom control, such as a toolbar button, is a specialized type of child window that responds to dialog messages and participates with event processing with other similar (button) windows.

Each child may talk to its parent. For instance, when one clicks a button with the mouse, a child window control sends a WM_COMMAND message to its parent window. The values of wParam and 1Param are as follows:

wParam	Child window ID
LOWORD (1param)	Child window handle
HIWORD (1param)	Notification code

The child window ID is the value passed to Windows CreateWindow when the child window is created; the child window handle is the value that windows returns from the CreateWindow call which created the child. The notification code is a submessage code which the child window uses to tell the parent window in more detail what the command message means. By tracking movement of the screen cursor (e.g., by processing Windows "mouse messages"), one may determine if the cursor has traversed an object boundary (i.e., entered a new window).

Referring now to FIG. 3C, the status window of the present invention will be described. Status window **340** comprises a single child window positioned substantially along one edge of the client area, preferably positioned along the inferior or bottom portion of the client area. It includes a single clipping region **343** equal to the client area of the child window. The status window **340** serves to display hint message **341**, which is described in further detail below.

As shown in FIG. 3D, a status window may include multiple clipping regions for the display of different messages. Status window **350**, for instance, includes a clipping region **353** for the display of hint text **351**. In addition, it includes a clipping region **355** for the display of state text **356**, and a clipping region **357** for the display of status text **358**. State text **356** indicates a state of a key of interest (e.g., caps lock key); status text **358**, on the other hand, indicates the status of the program (e.g., "READY" for a spreadsheet application).

The construction and operation of child windows in a windowing environment such as Microsoft Windows is known in the art. From Petzold's *Programming Windows*, for instance, a detailed description may be found in chapter 6 (Child Window Controls). For a discussion

specific to toolbars, see Ladd, S., *Creating Control Bars in Windows*, PC Techniques, October/November 1992, pages 32–36. The disclosures of each of the foregoing are hereby incorporated by reference.

In an alternate, more preferred method of the present invention, the toolbar and status window are created as embedded frames of the main window. In particular, the client area **310** of the window **300** is decreased in size by an amount equal to the area of the toolbar and the status window or frame. This technique has the advantage that events within either frame are processed by the windows procedure for the main window, thus simplifying design.

In a preferred embodiment, a buttons of a frame-based toolbar may be constructed and maintained from the following data structure (BUTTONDATA record):

```
// Internal representation of a button
typedef struct
{
    BUTTONINFO    info;     // Contain state and other
                            info about button
    HBITMAP       hBmp;     // Bitmap use to draw the button
    RECT          rc;       // Coordinate relative to
                            the desktop window.
} BUTTONDATA;
```

An array of BUTTONDATA records may be referenced through a pointer to the structure (after appropriate memory allocation):

```
typedef BUTTONDATA far * PBUTTONDATA;
```

As shown, each individual object or button may be tracked by storing a rectangle (pair of points) defining the boundary of the object. The toolbar, in turn, is constructed from a plurality of such buttons as follows (DESKBARINFO record).

```
// This is the main structure of the toolbar.
typedef struct
{
    HWND          hwndReside;    // current host
                                 of the toolbar
    PBUTTONDATA   pBtnList;      // Array of button
                                 (using
                                 // struct shown
                                 above)
    i16           nCount;        // Number of button
                                 in tool bar
    i16           idButtonHelp;  // String id of "hints"
                                 to show
                                 // in status.
} DESKBARINFO;
```

An array of DESKBARINFO records may be referenced through a pointer to the structure (after appropriate memory allocation):

```
typedef DESKBARINFO far * PDESKBARINFO;
```

In this manner, a plurality of toolbars, each having a plurality of button objects, may be constructed and maintained.

Improved User Feedback: "Hints"

Referring now to FIGS. 4A–C, the method of providing user interface "hints" in accordance with the present invention will now be introduced. As will be recalled from FIG. 3A, the toolbar **330** includes a plurality of bitmaps, each one indicating a particular action to be performed by the system upon user selection. As will be readily appreciated by inspecting the icons, the functionality associated with each is by no means self-evident. While the user may be able to discern certain functionality, such as a "cut" operation from a bitmap displaying a pair of scissors, by far the majority of icons will employ bitmap images which cannot be discerned upon first inspection. In prior art systems, therefore, a user would be forced to first "look up" the meaning of an icon before he or she uses it. Moreover, the user would have to then memorize the defined meaning (or resort to continually looking up its meaning). As a result, toolbars of prior art systems are typically under-utilized by most end users.

The method of the present invention for providing hints is perhaps best illustrated by an example. FIG. **4A** illustrates an application workspace **400** for a popular desktop database program, Paradox® for Windows, available from Borland International of Scotts Valley, Calif. The series of images (**400***a–d*) animates the operation of the user moving a screen cursor from an initial position **401** to a final position **407**. As the cursor sweeps from left to right, its tip ("hot spot") traverses several toolbar buttons. According to the present invention, as the cursor touches (i.e., enters the window of) a button, a descriptive message or "hint"—typically a string of text and/or graphic—is displayed in the status line window. As soon as the cursor moves away from a particular button, its corresponding hint is removed (erased) from the status window. Thus, in the example at hand, upon entry of the cursor into the first toolbar button (cursor position **401**), a descriptive hint **402** is displayed for the button. As shown by the text **402**, the user may now readily discern that the first button invokes the system command of "Open Table" (for opening a database table). In a similar manner, at cursor position **403**, hint **404** is displayed in the status window for indicating that the second button invokes the system "Open Form" command. At cursor position **405**, the hint "Open Query" **406** is displayed for the third button, and at cursor position **407**, the hint "Open Report" **408** is displayed indicating the function of the fourth button.

Continuing on to FIG. **4B**, the cursor continues its sweeping motion across the various toolbar buttons. As shown at **400***e*, the cursor position **410** elicits the "Open Script" hint **411**, indicating the function of the fifth button, and at cursor position **412**, the hint **413** is displayed to indicate that the sixth button invokes an "Open Library" command. Completing the example, upon entry of the cursor into the seventh button, at cursor position **414**, "Add Folder Item" hint **415** is displayed, and upon entry of the cursor into the last icon, at cursor position **416**, hint "Remove Folder Item" **407** is displayed. The toolbar **330** and its corresponding hints are summarized in FIG. **4C**.

As shown in FIG. **5A**, popular applications, such as Paradox for Windows, typically have a multitude of toolbars, each one being employed in a specific context of the system. The figure shows two screen objects **620** displayed within client area **610**. As Paradox is a database application, typical client area objects include a database table (e.g., customer.db) and form for that table (shown by the window **621**). As can be seen by toolbar **630**, the individual buttons comprising the toolbar have been updated for the current context at hand (creating a form for a database table).

The purpose of the example is twofold. First, from the location of the cursor at position **623**, it can be seen that the user interface hints of the present invention may be adapted for other screen objects (in this instance, a database form). Thus, at status window **640**, descriptive text **642** is displayed indicating the current relevance of the object which the cursor is touching.

Second, the example serves to illustrate that a user must typically deal with a multitude of toolbars, each having numerous bitmap images. Thus, for instance, toolbar **630** includes a plurality of different bitmap images, ones designed specifically for the task at hand. Again, in the small space afforded by a button object, the information conveyed is often insufficient to discern the meaning of the button object. According to the present invention, a different set of hints are provided for the toolbar **630**. As shown in FIG. **5B**, as the user sweeps the cursor from left to right, as animated by frames (a)–(p), the status window is updated with an appropriate hint. The Form Design toolbar **630** with its exemplary hints is summarized in FIG. **5C**.

Addition exemplary hints for a plurality of different toolbars is shown in FIGS. **6A–F**. As is readily apparent from the figures, a complex Windows application such as Paradox for Windows has far more iconic or bitmap images than could possibly be memorized by a typical user. Moreover, the function of any particular image may vary from one context to another. In a Report Design mode, for instance, a "lightning bolt" button may be defined to invoke a "View Data" operation (as shown in FIG. **6A**). That very same image in a database query mode, on the other hand, may assume an entirely different meaning, such as "Run Query" (as shown in FIG. **6B**). Thus, the present invention preserves the interface advantage of employing a toolbar and, at the same time, eliminates the profound disadvantage attendant with toolbars.

Internal Operation

Referring now to FIG. **7**, a hint method **700** of the present invention will be described by means of a flowchart. The method proceeds as follows. As shown by the move mouse event **701**, the method is an event-driven process, with screen cursor or "mouse messages" being of particular interest. Upon receipt of a mouse move message at **701**, the method compares the current cursor position (as reported with the mouse messages) against a list of known UI objects. In the instance where the UI object of interest is a toolbar, for example, the cursor position may be compared against a list of rectangles (Windows Rect structures). In this manner, entry of the cursor within an object of interest (i.e., the "hot spot" of the cursor touching the object) may be readily determined. At step **703**, if the screen cursor enters an object of interest (yes), then at step **705** the hint (text string) for that object is determined (e.g., from the Rect list), and at step **705** the hint text is sent to the status window for display. If, on the other hand, the mouse has not entered an object (no at step **703**), then the system may undertake other activity, as indicated by step **711**.

Upon entry of the cursor into the object, the status line continues to display the hint until the cursor departs that object; this is indicated by steps **706** and **707**. In particular, if the screen cursor departs the object at step **706** (yes), then at step **707**, the previously-displayed hint text is erased from the status window. If, on the other hand, the cursor has not departed (no at step **706**), then the system may undertake other activity, as indicated by step **713**. Upon the cursor's departure and accompanying erasure of hint text, the method awaits the next user event (conceptually loops back to step **701** to await the cursor's entry into the next child window).

As shown by the figure, the method also responds to a timer event **710**. In the preferred embodiment where mouse move messages are processed by the main window's window procedure, it is possible for the main window to "miss" a move mouse message. A user may, for instance, rapidly move the cursor out of the main window. In such a case, new move mouse messages are directed to the queue of the destination window, thus causing the main window to "miss" movement of the cursor out of a particular object being described with a hint. By periodically refreshing the hint, typically by setting a timer (for generating timer messages), this situation may be avoided. In particular, a window which has lost the current focus and "missed" the departure of the cursor will query itself for reconciling any differences between the current cursor location and the hint being displayed.

The following code snippet includes exemplary processing of the mouse event and timer event from the desktop window procedure.

```
U32 WNDPROC DeskWndProc( HWND hwnd,
U16 msg, U16 wParam, U32 lparam )
{
    PWINDATA           pData;
    PDESKBARINFO       pDeskBar;
    i16                count;
    PBUTTONDATA        pBtn;
    I32                ret;
    // Data for tool bar attached to instance of application
    window.
    pData = (PWINDATA) GetWindowWord( hwnd,
    WINDATAOFFSET );
    switch (msg)
    {
         .
         .
         .
    //
    // Process mouse event from the desktop window.
    //
        case WM_MOUSEMOVE:
            // Retrieve deskbar info
            pDeskbar = pData->DeskBar;
            // Keep current mouse coordinate.
            ptMouse = MAKEPOINT( lparam );
            pBtn = NULL;
            // Hit test process: Walk list of button in toolbar
            // until we found one hit by the mouse.
            for ( count = pDeskBar->nCount, pBtn =
            pDeskBar->pBtnList;
                count;
                count--, pBtn++)
                {
                if (PtInRect(&pBtn->rc, ptMouse))
                    break;
                }
            // If mouse over a button, show help in status
            // otherwise clear status line
            if (pBtn)
                DisplayHelpString(pDeskBar, pBtn->info.id);
            else
                ClearStatusLine();
            break;
    //
    // Retrieve timer message to clear status
    line message when mouse is
    // not over a button and possibly over a child window.
    //
        case WM_TIMER:
            // Retrieve deskbar info
            pDeskbar = pData->DesBar;
```

-continued

```
            // Get current mouse coordinate ourself.
            GetCursorPos(&ptMouse);
            pBtn = NULL;
            // Hit test process: Walk list of button in toolbar
            // until we found one hit by the mouse.
            for ( count = pDeskBar->nCount,
            pBtn = pDeskBar->pBtnList;
                count;
                count--, pBtn++)
                {
                if (PtInRect(&pBtn->rc, ptMouse))
                    break;
                }
            // If mouse not over a button, clear status line
            if (!pBtn)
                ClearStatusLine();
            break;
         .
         .
         .
    }
```

Referring now to FIG. **8**, adaptation of the hint method of the present invention for a screen object with multiple regions will now be described. FIG. **8** shows a screen object **801** having an upper and a lower half. Upon movement of the screen cursor to the upper half, the hint "Ascending SpeedSort button" **803** is shown. Upon movement of the cursor to the lower half of the object, the hint "Descending SpeedSort button" **805** is displayed. The foregoing hit-testing technique may be employed for determining subregions of an object, for example, by dividing the object into upper and lower reference rectangles. In this manner, the method of the present invention may easily be adapted to objects having multiple regions of interest.

Advantages Over Prior Art

The present invention recognizes that conventional techniques for providing screen feedback or help, such as "balloon help" and "intelligent cursors," operate in a fashion which is not only intrusive to the active workspace (client area), but is also computationally very inefficient. Balloon help, for instance, by "popping up" at a screen location proximate the cursor location, typically obscures the work product of interest. Moreover, the task of continually creating and destroying user interface windows (the "balloons"), being resource intensive, substantially degrades the performance of such systems. By overwriting the client area (i.e., active screen portion where the user's data is represented), for instance, these systems destroy the screen image of the underlying work product. To repair this image (upon cessation of the balloon help), such systems must "repaint" the image, typically by either regenerating the image from underlying data structures or restoring a prior-saved bitmap (portion overwritten). In either instance, performance of the system suffers as these are resource intensive operations.

The present invention eschews these "balloon help" or "intelligent prompting" methods. Instead, the hint prompts of the present invention are confined to a screen region removed from the user's main focus of attention. In this manner, the hint method of the present invention operates transparently, that is, it does not distract the user with help information which is not needed. Instead, if the user requires additional information for an object of interest, the user need only glance down at the screen region reserved for the hint. Balloon help systems, in contrast, are very distracting to use, as balloon windows are continually popping up in the screen region of main interest to the user. Moreover, since such systems are relatively resource (computational) intensive, the screen cursor often appears to "hang" while such systems are busy creating and destroying balloon objects. Thus, the present invention is advantageous not only in terms of user interface design, but also advantageous in its efficient use of system resources.

While the invention is described in some detail with specific reference to a single preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. Thus, the true scope of the present invention is not limited to any one of the foregoing exemplary embodiments but is instead defined by the following claims.

What is claimed is:

1. In a computer system having a screen device, a method for providing a user with information about pictorial graphical icons representing user-selectable commands displayed on the screen, the method comprising:

(a) displaying a screen cursor on the screen device to indicate location;

(b) moving the screen cursor to a screen location of interest;

(c) if the screen cursor touches one of the pictorial graphical icons, displaying at a display location substantially along one side of the screen device a message describing said pictorial graphical icon;

(d) moving the screen cursor to a second screen location of interest;

(e) if the screen cursor touches a second one of the pictorial graphical icons, displaying a second message describing said second pictorial graphical icon at said display location; and

(f) independently of steps (b)-(e), refreshing said information provided to the user about pictorial graphical icons by periodically testing whether the screen cursor still is positioned at a pictorial graphical icon whose derivative message is being displayed.

2. The method of claim **1**, wherein said display location occupies the bottom-most portion of the screen.

3. The method of claim **1**, wherein step (b) includes moving a pointing device which is operably coupled to the screen cursor.

4. The method of claim **1**, wherein step (c) includes:

storing a list of regions for pictorial graphical icons of interest;

storing a descriptive message for each said pictorial graphical icon of interest;

determining from the list of stored regions whether the cursor lies within one of the pictorial graphical icons of interest; and

if the cursor is found to lie within one of the pictorial graphical icons, displaying the descriptive message for that pictorial graphical icon.

5. The method of claim **1**, wherein said message includes text information describing the pictorial graphical icon.

6. The method of claim **1**, wherein said message includes graphic information describing the pictorial graphical icon.

7. The method of claim **1**, further comprising:

(g) moving the screen cursor away from the pictorial graphical icon; and

(h) erasing the message.

8. The method of claim 7, further comprising:

repeating steps (b)-(d) for a plurality of screen locations, whereby a descriptive message is displayed for each pictorial graphical icon the screen cursor touches.

9. The method of claim **1**, wherein the display location includes a small display window positioned away from the user's main focus of attention.

10. The method of claim 1, wherein said display location is positioned sufficiently far away from a pictorial graphical icon being touched by the cursor so that the user cannot perceive the message while looking at the pictorial graphical icon.

11. A graphical user interface system comprising:

a computer having a screen device for displaying pictorial graphical icons representing system commands;

means for indicating a location on the screen device with a cursor;

means for determining if the cursor lies within one of the pictorial graphical icons;

means for displaying a descriptive message when the cursor lies within one of the pictorial graphical icons, said message being located substantially along one edge of the screen device;

means for changing said message when said cursor is moved from one pictorial graphical icon to another; and

refresh means, operating independently of said determining means, for periodically testing whether the cursor still is positioned at a pictorial graphical icon whose descriptive message is being displayed.

12. The system of claim **11**, wherein said means for determining includes:

means for storing a location for desired ones of the pictorial graphical icon on the screen device, and determining if the cursor is positioned at one of the stored locations.

13. The system of claim **11**, wherein said message includes selected combinations of text and graphic information.

14. The system of claim **11**, wherein said one edge is a bottom-most edge of the screen device.

15. The system of claim **11**, wherein said one edge is located substantially away from the user's main focus of attention, and wherein said message is perceived only when the user looks directly at the edge.

16. The system of claim **11**, wherein the pictorial graphical icons displayed on the screen device include both text and graphic screen objects.

17. The system of claim **11**, wherein the pictorial graphical icons are displayed on the screen device in a toolbar having a plurality of pictorial graphical icons, each of said pictorial graphical icons comprising a non-textual image for representing a system command to be performed upon selection of one of said pictorial graphical icons of said toolbar with the cursor.

18. The system of claim **11**, wherein said means for displaying a descriptive message includes:

means for assigning selected ones of said pictorial graphical icons an identifier for identifying a pictorial graphical icon together with textual information describing the pictorial graphical icon;

means, responsive to said determining means, for matching an assigned identifier with a pictorial graphical icon which the cursor is positioned at; and means for identifying the textual information stored with the matched assigned identifier.

19. The system of claim **11**, wherein said indicating means includes a pointing device for positioning the cursor at desired screen locations in response to signals generated by the pointing device upon movement by a user, and wherein said determining means operates in response to generation of said signals.

20. In a computer system including a display screen having a multi-window graphical user interface with a screen cursor, said multi-window graphical user interface including at least one pictorial graphical screen icon for invoking a user command upon selection with the screen cursor, said multi-window graphical user interface having application programs operating in response to messages posted by the system to said application programs, said messages for notifying said application programs of occurrence of events in the system, a method for providing messages to a user describing a command of a pictorial graphical screen icon of interest, the method comprising:

(a) for each said at least one pictorial graphical screen icon, storing a message describing the pictorial graphical screen icon;

(b) displaying one of said application programs on the display screen, the displayed application program having a main window and a status window for displaying said messages;

(c) positioning the screen cursor at a desired one of said pictorial graphical screen icons;

(d) upon entry of the screen cursor within a boundary of the desired pictorial graphical screen icon, identifying which one of the stored messages describes the desired pictorial graphical screen icon, said entry being detected by the displayed application program in response posting of a cursor movement message by the system upon movement of said screen cursor;

(e) displaying the identified message within the status window;

(f) repeating steps (c)-(e) for a plurality of pictorial graphical screen icons, whereby the status window is continually updated with a message describing a pictorial graphical screen icon currently pointed to by said screen cursor; and

(g) independently of steps (d) and (e), periodically refreshing said status window by:

(i) setting a system timer for generating periodic timing signals; and

(ii) upon occurrence of a timing signal, refreshing the status window by determining if said screen cursor is still positioned within any of said at least one pictorial graphical screen icon and erasing any message which is displayed within said status window if said screen cursor is not positioned within a pictorial graphical screen icon, so that a correct message is displayed within said status window irrespective of whether a cursor movement message has been posted by the system in response to movement of said screen cursor.

21. The method of claim **20**, wherein each said at least one pictorial graphical screen icon comprises a bit-mapped image occupying a relatively small screen area of the display screen.

22. The method of claim **20**, wherein each of the messages includes textual information describing a command in detail sufficient to assist a novice user.

23. The method of claim **20**, further comprising:

(h) upon exit of the screen cursor from the desired pictorial graphical screen icon, removing the displayed message from the status window.

24. The method of claim **20**, wherein said status window is positioned on the display screen at a location fixed relative to the user interface.

25. The method of claim **20**, wherein said status window is positioned on the display screen at a location which is removed from a main focus of attention of the user.

26. The method of claim **20**, wherein said status window includes first and second clipping regions, and wherein said first clipping region includes messages describing commands of pictorial graphical screen icons, and wherein said second clipping region includes messages describing a general status of the system.

27. The method of claim **20**, wherein each of said at least one pictorial graphical screen icon is rectangular in shape, and wherein step (d) includes:

comparing a current position of the screen cursor to a list of known rectangles, each of said known rectangles representing a boundary of a pictorial graphical screen icon; and

if the screen cursor is positioned within a boundary of a pictorial graphical screen icon, retrieving the stored message for the pictorial graphical screen icon.

28. The method of claim **20**, wherein each of said at least one pictorial graphical screen icon is located on the display screen with an orientation which is constant relative to one another and constant relative to the status window.

29. The method of claim **20**, wherein said multi-window graphical user interface includes a plurality of window-based programs, each said window-based program comprising a main window having a client area for displaying data objects created by the user, and wherein said status window is positioned within the main window of one of said window-based programs at a location removed from the client area for that window-based program.

30. The method of claim **20**, wherein said user interface includes a main window for displaying screen objects generated with application software, and wherein said status window is positioned along an edge of the main window.

31. In a computer system having a display screen with a user interface, said user interface including a toolbar comprising at least one pictorial graphical screen button, a method for providing hints to the user about the function of a pictorial graphical screen button of interest, the method comprising:

(a) for each said at least one screen button, storing a hint message together with an identifier indicating a screen button associated with the stored hint message, said pictorial graphical screen button invoking a system function upon selection by a user wherein said system function is selected from a group consisting of cutting, pasting, editing, and printing functions;

(b) moving a screen cursor for selecting a desired position on the display screen;

(c) upon movement of said screen cursor, determining if said screen cursor touches one of said at least one pictorial graphical screen button;

(d) if said screen cursor touches one of said at least one pictorial graphical screen button, identifying the one pictorial graphical screen button touched;

(e) displaying for each pictorial graphical screen button identified its hint message at a position on the display screen fixed relative to the user interface; and

(f) independently of steps (b)–(d), refreshing said hints provided to the user by periodically testing whether said screen cursor touches a pictorial graphical screen button whose hint message is being displayed.

32. The method of claim **31**, further comprising:

(g) repeating steps (c)–(e) for a plurality of screen locations, whereby a plurality of pictorial graphical screen buttons are touched by the screen cursor and are described to the user by the system.

33. The method of claim **31**, further comprising:

if said screen cursor does not touch a pictorial graphical screen button in step (d), erasing any hint message which is displayed.

34. The method of claim **31**, wherein said user interface includes a region for displaying objects created by the user, and wherein said hint messages are displayed on the display screen at a location other than said region.

35. The method of claim **31**, wherein selected ones of said at least one pictorial graphical screen button are clustered together.

36. A system for assisting a user with operation of a computer, the system comprising:

a computer having a processor and a memory;

a screen device for displaying a multi-window graphical user interface in first, second, and third display regions, said first display region for displaying objects having data supplied by the user, said second display region for displaying non-textual, pictorial icons representing operations to be performed on said objects by the computer upon selection by the user, and said third display region for providing information assisting the user with understanding a non-textual, pictorial icon;

pointing means for positioning a screen cursor at a desired screen location on the screen device;

means, responsive to said pointing means, for determining each one of said non-textual, pictorial icons which is traversed by the screen cursor;

means for describing to the user at the third display region each non-textual, pictorial icon as it is traversed by the screen cursor; and

update means, operating independently of said means for determining, for repetitively testing a current screen location of the screen cursor for determining whether the screen cursor is still traversing a non-textual, pictorial icon which is being described at the third display region.

37. The system of claim **36**, wherein the describing means includes means for storing in the memory descriptive information for each of said non-textual, pictorial icons.

38. The system of claim **36**, wherein the screen device includes a multi-window graphical user interface having a client area and a status line, and wherein said first display region includes said client area and wherein said second display region includes said status line.

39. The system of claim **38**, wherein said status line is located along a peripheral portion of said multi-window graphical user interface.

40. The system of claim **36**, wherein said pointing means includes a mouse device for generating signals upon movement of the device by the user, and wherein said determining means is responsive to occurrence of said signals.

* * * * *

Exhibit 10

US005544360A

United States Patent [19]

Lewak et al.

[11] **Patent Number: 5,544,360**

[45] **Date of Patent: Aug. 6, 1996**

[54] **METHOD FOR ACCESSING COMPUTER FILES AND DATA, USING LINKED CATEGORIES ASSIGNED TO EACH DATA FILE RECORD ON ENTRY OF THE DATA FILE RECORD**

[75] Inventors: **Jerzy Lewak**, Del Mar; **Slawek Grzechnik**, La Mesa; **Jon Matousek**, San Diego, all of Calif.

[73] Assignee: **Paragon Concepts, Inc.**, Solana Beach, Calif.

[21] Appl. No.: **384,379**

[22] Filed: **Feb. 3, 1995**

Related U.S. Application Data

[63] Continuation of Ser. No. 980,620, Nov. 23, 1992, abandoned.

[51] **Int. Cl.**^{6} **G06F 17/30**

[52] **U.S. Cl.** **395/600**; 364/253; 364/253.3; 364/282.1; 364/282.3; 364/283.1; 364/283.2; 364/283.3; 364/286.4; 364/286.5

[58] **Field of Search** 364/282.1, 282.3, 364/283.1, 283.2, 283.3, 286.4, 286.5

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,047,918	9/1991	Schwartz et al.	364/200
5,115,504	5/1992	Belove et al.	395/600
5,162,992	11/1992	Williams	364/419
5,201,047	4/1993	Maki et al.	395/600
5,201,048	4/1993	Coulter et al.	395/600
5,204,947	4/1993	Bernstein et al.	395/157
5,206,949	4/1993	Cochran et al.	395/600
5,276,867	1/1994	Kenley et al.	395/600

OTHER PUBLICATIONS

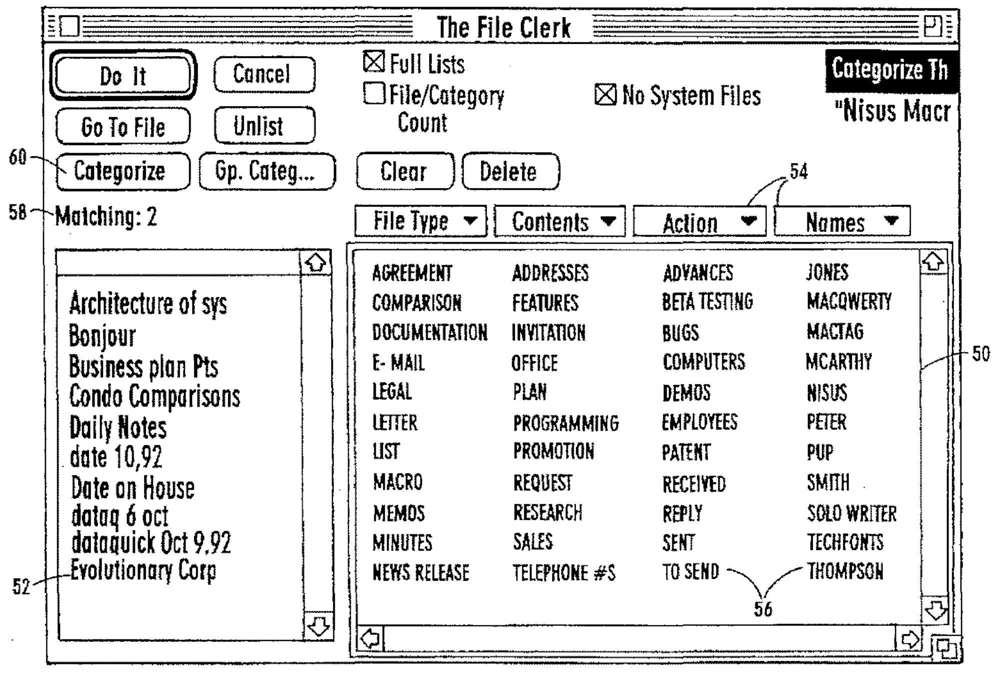
Van Kir., "Lotus, Traveling Software Tackle DOS File Management Problems", Apr. 1989, V2R4 p. 35(2) PC-Computing.
PC Tools DOS & Shell/File Manager, 1991, pp. 33–35, (Version 7 For DOS) Central Point Software Inc.
Central Point Software Inc. PC Tools Data Recovery And Systems Utilties, 1991, pp. 95–97 (Ver. 7 For DOS).
Thought Pattern Handbook, By Bananafish Software, Inc., 1991 San Francisco, CA.

Primary Examiner—Thomas G. Black
Assistant Examiner—Jack M. Choules
Attorney, Agent, or Firm—Fish & Richardson P.C.

[57] **ABSTRACT**

A computer filing system for accessing files and data according to user-designated criteria. The system allows the user to define a virtually unlimited number of hybrid folders by describing, using terms of their own selection, the file contents of those files which are to belong to particular hybrid folders. Such hybrid folders can be implemented on top of, and used in addition to, the normal hierarchical structured directory, or they may replace such normal structures entirely. The inventive computer file control system could therefore be used as the basis of a new computer operating system. In the process of search and retrieval, the invention ensures in two ways that the user defines a filter which will always find at least one file. The user is not required to type the key words to search but instead chooses the words from pick lists, making mistyping impossible. As the user builds the search filter definition, categories which would find no data are automatically excluded as pick list possibilities.

21 Claims, 4 Drawing Sheets



Architecture of sys
Bonjour
Business plan Pts
Condo Comparisons
Daily Notes
date 10,92
Date on House
dataq 6 oct
dataquick Oct 9,92
Evolutionary Corp

File Type	Contents	Action	Names
AGREEMENT	ADDRESSES	ADVANCES	JONES
COMPARISON	FEATURES	BETA TESTING	MACQWERTY
DOCUMENTATION	INVITATION	BUGS	MACTAG
E-MAIL	OFFICE	COMPUTERS	MCARTHY
LEGAL	PLAN	DEMOS	NISUS
LETTER	PROGRAMMING	EMPLOYEES	PETER
LIST	PROMOTION	PATENT	PUP
MACRO	REQUEST	RECEIVED	SMITH
MEMOS	RESEARCH	REPLY	SOLO WRITER
MINUTES	SALES	SENT	TECHFONTS
NEWS RELEASE	TELEPHONE #S	TO SEND	THOMPSON

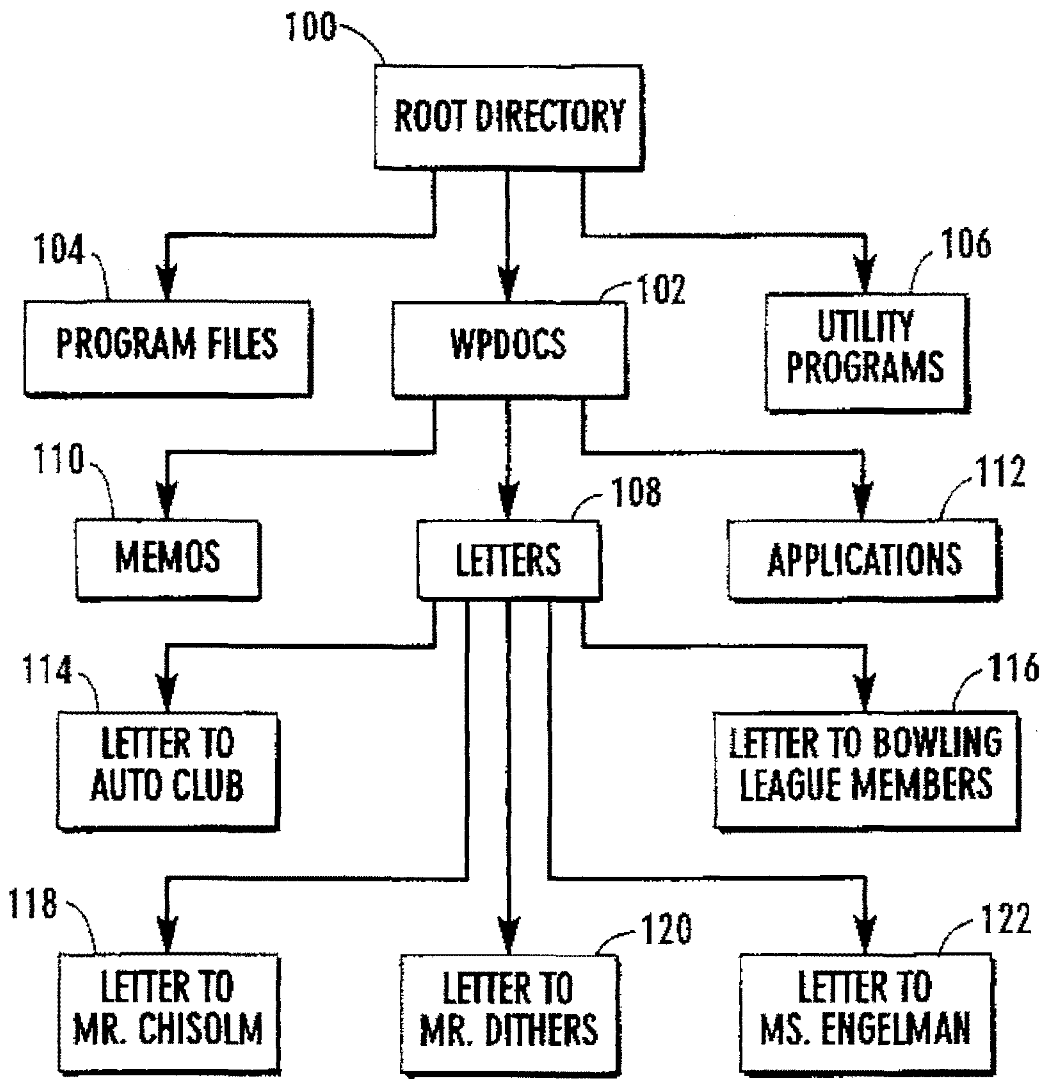
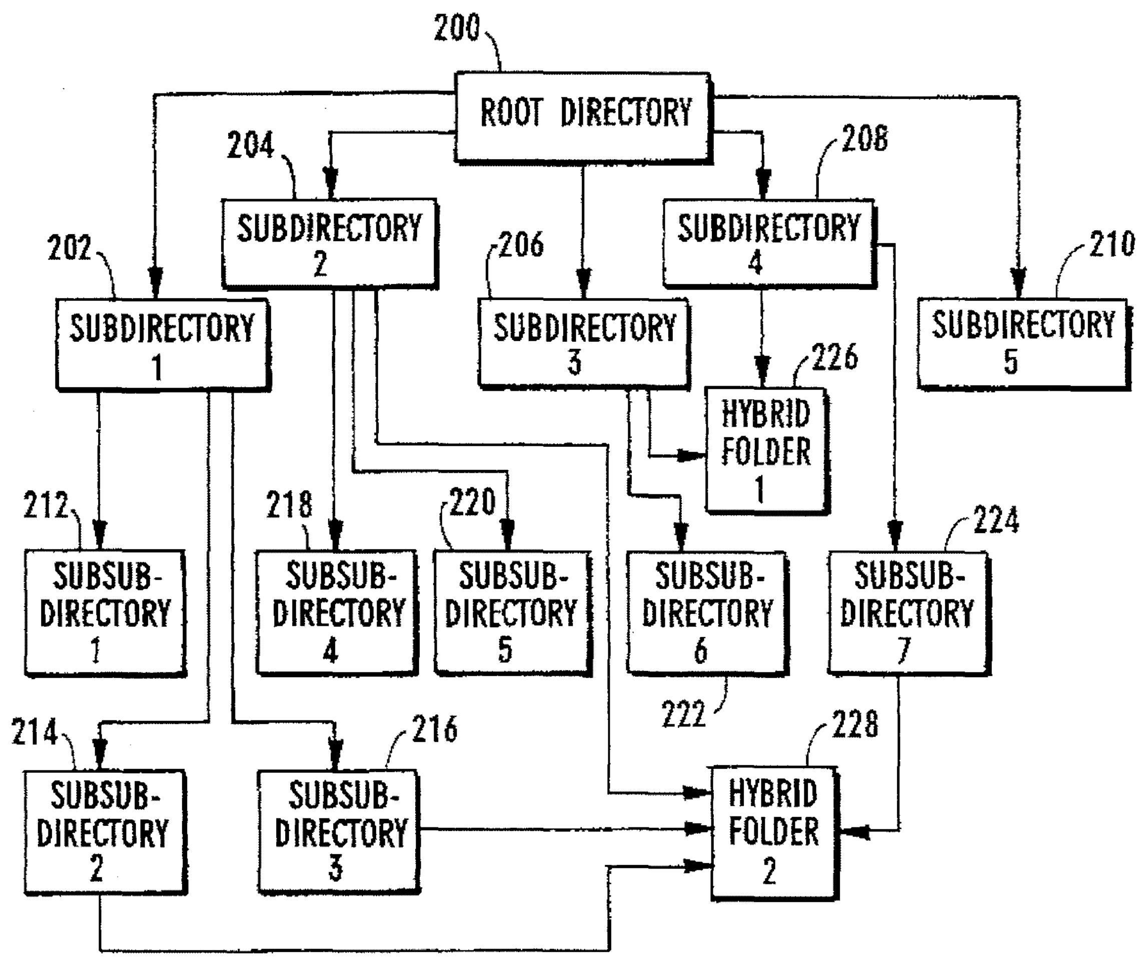


FIG. 1
(PRIOR ART)

FIG. 2

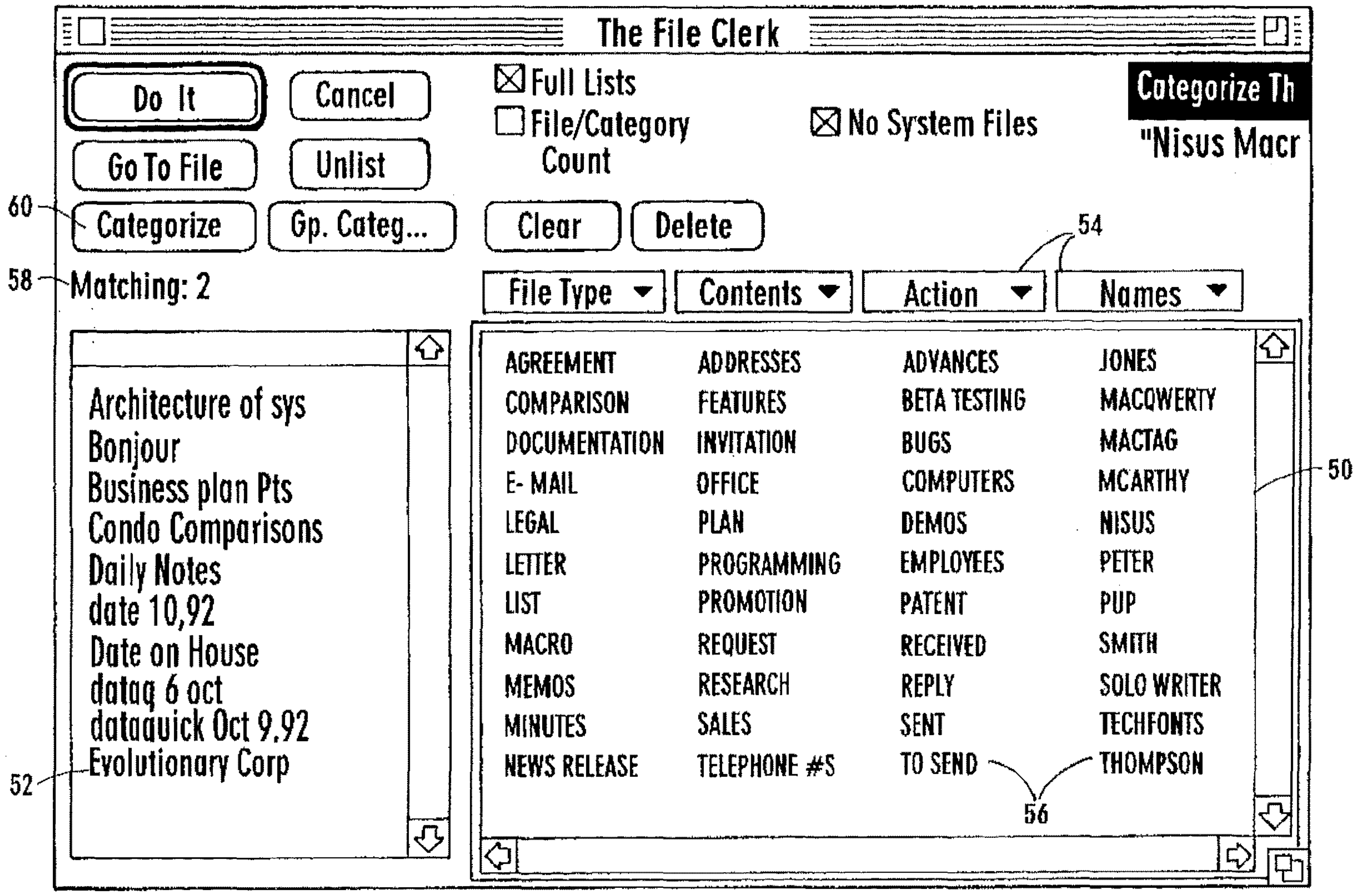

200
ROOT DIRECTORY
204
208
202
SUBDIRECTORY 2
206
SUBDIRECTORY 4
210
SUBDIRECTORY 1
SUBDIRECTORY 3
SUBDIRECTORY 5
226
HYBRID FOLDER 1
212
218
220
224
SUBSUB-DIRECTORY 1
SUBSUB-DIRECTORY 4
SUBSUB-DIRECTORY 5
SUBSUB-DIRECTORY 6
SUBSUB-DIRECTORY 7
222
214
216
228
SUBSUB-DIRECTORY 2
SUBSUB-DIRECTORY 3
HYBRID FOLDER 2

FILE_TYPE	CONTENTS	ACTION	NAMES
000 AGREEMENT	100 ADDRESSES	200 ADVANCES	300 JONES
001 COMPARISON	101 FEATURES	201 BETA TESTING	301 MACQWERTY
002 DOCUMENTATION	102 INVITATION	202 BUGS	302 MACTAG
003 E- MAIL	103 OFFICE	203 COMPUTERS	303 MCARTHY
004 LEGAL	104 PLAN	204 DEMOS	304 NISUS
005 LETTER	105 PROGRAMMING	205 EMPLOYEES	305 PETER
006 LIST	106 PROMOTION	206 PATENT	306 PUP
007 MACRO	107 REQUEST	207 RECEIVED	307 SMITH
008 MEMOS	108 RESEARCH	208 REPLY	308 SOLO WRITER
009 MINUTES	109 SALES	209 SENT	309 TECHFONTS
010 NEWS RELEASE	110 TELEPHONE #S	210 TO SEND	310 THOMPSON
•	•	•	•
•	•	•	•
•	•	•	•

FIG. 3

FILE_NAME	FILE_LOC	DATE/TIME	NO_CATEGORIES	CATEGORY_ARRAY
jones.mem	c:\memos	01-01-80 01:30	2	008, 300
minutes.1	f:\wpdocs	10-10-90 10:10	3	009, 103, 109
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

FIG. 4



Architecture of sys
Bonjour
Business plan Pts
Condo Comparisons
Daily Notes
date 10,92
Date on House
dataq 6 oct
dataquick Oct 9.92
Evolutionary Corp

AGREEMENT	ADDRESSES	ADVANCES	JONES
COMPARISON	FEATURES	BETA TESTING	MACQWERTY
DOCUMENTATION	INVITATION	BUGS	MACTAG
E- MAIL	OFFICE	COMPUTERS	MCARTHY
LEGAL	PLAN	DEMOS	NISUS
LETTER	PROGRAMMING	EMPLOYEES	PETER
LIST	PROMOTION	PATENT	PUP
MACRO	REQUEST	RECEIVED	SMITH
MEMOS	RESEARCH	REPLY	SOLO WRITER
MINUTES	SALES	SENT	TECHFONTS
NEWS RELEASE	TELEPHONE #S	TO SEND	THOMPSON

FIG. 5

METHOD FOR ACCESSING COMPUTER FILES AND DATA, USING LINKED CATEGORIES ASSIGNED TO EACH DATA FILE RECORD ON ENTRY OF THE DATA FILE RECORD

This is a continuation of application Ser. No. 07/980,620 filed on Nov. 23, 1992, now abandoned.

NOTICE OF COPYRIGHTS

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to computer file control systems, and more specifically to a flexible system for accessing computer files and data therein according to user-designated criteria.

2. Description of Related Art

Many currently-existing computer file control systems employ a hierarchical filing structure. This system emulates commonly-used paper filing systems, but is more general and is capable of more extensive use. Thus, for example, in a paper filing system, a filing cabinet may contain 4 or 5 drawers. Each drawer may contain perhaps a dozen or so hanging files and each file may contain two or three file folders. Typically, a hanging file can only hold a very small number of file folders before it becomes unmanageable, so that a new hanging file needs to be started.

A typical computer system organizes data into files (analogous to papers in a paper filing system) and directories (analogous to file folders and hanging files). Indeed, in graphically-oriented computer systems, directories are sometimes known as "folders." Directories may contain other directories (also referred to as subdirectories) and files.

FIG. **1** shows a simplified typical hierarchical tree-type directory structure. This structure is called "tree" because it looks like an upside-down tree, with the base, or "root" of the tree at the top. Subdirectories are often referred to as "branches" of the tree, and files are often referred to as leaves of the tree.

In FIG. **1**, the root directory **100** contains a number of subdirectories **102–112** and files **114–122**. The subdirectories **102–112** may contain other subdirectories and files, and so on.

In typical use, directories often contain files having similar kinds of data. Also, the name of the directory is typically selected to be descriptive of the kinds of files and directories therein. For example, a WPDOCS directory **102** might contain wordprocessing documents and directories for holding specific categories of such documents. For example, a LETTERS directory **108** may contain only files which are letters: "LETTER TO AUTO CLUB" **114**, "LETTER TO BOWLING LEAGUE MEMBERS" **116**, "LETTER TO MR. CHISOLM" **118**, "LETTER TO MR. DITHERS" **120**, and "LETTER TO MS. ENGELMAN" **122**. Memos could be stored in a "MEMOS" subdirectory **110**, patent applications in an "APPLICATION" subdirectory **112**, etc.

Custom structures of such directories are created so as to make the storing and retrieval of files convenient. If the number of files to be stored is small and the number of different file kinds is either small or very well defined, this type of file storage structure works well. However, several problems arise when the number of files becomes large, or if the file categories are not well-defined. In such cases, the hierarchical filing structure becomes very cumbersome to use for the following reasons:

1. The tree becomes quite deep, and so it takes more time to get to the end of any branch.
2. It becomes more difficult for the user to decide where to store a particular file.

As a result, finding particular files becomes harder and harder. Frequently, the user is not able to clearly and unambiguously associate a desired file with any one directory. In fact, the user often associates a file with several subdirectories. This happens both when a file is being saved and when it is being retrieved. The same problem arises in paper filing systems. Quite often a document may logically belong within many different folders, with the result that it is difficult to find a desired document once the document has been filed.

Clearly, a hierarchical topic-oriented file structure is too rigid for many applications where information must be organized into files. It has been found that users generally think in terms of overlapping categories or descriptions of file content, rather than in very strictly-defined, non-overlapping topics.

For example, referring now to FIG. **2**, a root directory **200** has five subdirectories **202–210**. In this example, subdirectory **202** has three "sub"-subdirectories **212–216**, subdirectory **204** has two "sub"-subdirectories **218–220**, subdirectory **206** has one "sub"-subdirectory **222**, and subdirectory **208** has one "sub"-subdirectory **224**. A file might logically belong in subdirectory **204**, sub-subdirectory **214**, sub-subdirectory **216**, and sub-subdirectory **224**. Similarly, another file might logically belong in subdirectory **206** and subdirectory **208**.

Therefore, what is really needed are "hybrid" logical directories or folders, which contain those files whose content overlaps more than one physical directory. Thus, in the present example, a first hybrid logical directory **226** would contain each file that logically belongs in subdirectory **206** and subdirectory **208**. Similarly, a second hybrid logical directory **228** would contain each file that logically belongs in subdirectory **204**, sub-subdirectory **21 4**, sub-subdirectory **21 6**, and sub-subdirectory **224**.

In the typical hierarchical directory structure illustrated in FIG. **1**, "hybrid" directories are not possible. Thus, it is very desirable to provide a method for accessing files consonant with the way users think of them, and not limited to how such files are stored in the computer.

Some systems have been developed to overcome the rigidity of typical hierarchial directory structures, but these systems have limitations. In one such method, all the words in a file are indexed and a concordance list associated with each file is created. The user then may search for files by file word content by defining "search filters", which are search terms in logically defined combinations (e.g., a search filter comprising "Smith" AND "tooling" would locate all documents having both the word "Smith" and the word "tooling"). However, this indexing method has the following shortcomings:

1. The words inside a document often do not identify the type of document. For example, a document which is a letter generally does not contain the word "letter".

Thus, a search for all documents that are letters will only identify documents which contain the word "letter". Similarly, it is very difficult to identify words which occur in all letters. Furthermore, this technique requires the user to remember precise words appearing in the file. This may be especially difficult for older files for which the user cannot recall the precise contents.

2. In applications where such a method might be most useful, the search is very time consuming. If the number of files is very large, the concordance list also is very large. A partial solution to this has been to have the program continually work in the background updating the concordance list as changes are made in the files. However, this process generally disrupts a user's activity.
3. Many files (such as binary files) contain information which is in a form that is not easily interpreted by the human mind. Such files cannot be stored and retrieved by these methods.
4. In preparing a search filter, a user must type the word or words targeted in the search. In such situations, the user commonly mistypes or use a different inflection, spelling or grouping of the key words. Without using the precise words as they appear in the files, a search has an even lower probability of success.
5. Because a user may change the contents of documents, the index of words must be constantly updated. This process is time consuming and distracting. In contrast, the topical description of a document changes very little, if at all, during its lifetime.

Because of these problems, the above methods are generally used in only "last resort" searches. Thus, the user is left to negotiate the hierarchical directory structure.

Databases and outlining programs also provide methods for data storage, retrieval, or re-organization. Databases may be created using relational techniques. In relational databases, the relationships between data are typically included in the database structure. Searches in a relational database may be made by creating a search of the relations. However, database searches are usually restricted in two ways: by the field of each data element and by the content of each field. In outlining programs and other thought-organization programs, the data is generally required to be hierarchically organized at the time of data entry.

Accordingly, it would be desirable to provide a method for accessing files which provides intuitive access by user-defined topics. Such a solution should provide: easy access to a large number of files and to files having overlapping categories; simple access to files stored in a hierarchical file system without the necessity of sorting through multiple levels; access to files using predefined categories descriptive of the contents of the files; access to files which permits a user to create a search filter of categories of files using precise category names to which the files belong; and a method of accessing files which is unaffected by changes in file contents.

The present invention provides such a method.

SUMMARY OF THE INVENTION

The invention comprises a computer file control system, with a suitable user interface, implemented as a software program, which allows total freedom from the restrictions imposed by hierarchical and other present day computer filing systems.

The invention allows a user to define categories for files stored in a computer system, and to edit such categories as

they are used, to designate all applicable categories for each file, and to link categories in user-definable ways. The invention further allows a user to be reminded of linked categories.

In the process of search and retrieval, the invention overcomes the problem of search filter definition, ensuring that the user defines a filter which will always find at least one file, thus avoiding wasting time in searching for data that cannot be matched. This is achieved in two ways. First, the user is not required to type the key words to search; instead, the user simply chooses the words from pick lists, making mistyping impossible. Second, as the user builds the search filter definition, categories which would find no data are automatically excluded as pick list possibilities.

More particularly, the invention allows users to define an unlimited number of their own "hybrid folders" by simply describing, using categories the user defines, the file contents of those files which are to belong to each "hybrid folder". This description is dynamic (that is, changeable by the user from time to time), and may be either totally unrestricted or restricted to a particular directory or subdirectory, as the user chooses. Such hybrid folders can be implemented on top of, and used in addition to, the normal hierarchical structured directory, or they may replace such normal structures entirely. The inventive computer file control system could therefore be used as the basis of a new computer operating system.

The details of the preferred embodiment of the present invention are set forth in the accompanying drawings and the description below. Once the details of the invention are known, numerous additional innovations and changes will become obvious to one skilled in the art.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a tree diagram illustrating the organizational structure of a typical prior art computer filing system.

FIG. 2 is a tree diagram illustrating the organizational structure of the computer filing system of the present invention.

FIG. 3 shows an example of a File Category Table in accordance with the present invention.

FIG. 4 shows an example of a File Information Directory in accordance with the present invention.

FIG. 5 shows an example of a Categories Window and File Window in accordance with the present invention.

Like reference numbers refer to like elements in the drawings.

DETAILED DESCRIPTION OF THE ILLUSTRATED EMBODIMENTS

Throughout this description, the preferred embodiment and examples shown should be considered as exemplars, rather than as limitations on the present invention.

The present invention consists of a computer file control system the includes a File Category Table ("FCT") and a File Information Directory ("FID") to store information about user-defined categories and information linking such categories to specific files. The invention uses the information stored in the FCT and FID to quickly and easily access files in the file system. (The term "file" should be understood to mean any collection of data or information stored on a computer system). In the preferred embodiment, the invention includes a graphical user interface for defining categories, associating files with particular categories, and defining search filters.

File Category Table Structure

In the preferred embodiment, the FCT is a table that can be modeled as having a set of columns labeled by category-type (e.g., FILE_TYPE, CONTENTS, ACTION, etc.), with entries comprising lists of category names or descriptions (e.g., the FILE_TYPE column might have entries for AGREEMENTS, E-MAIL, MEMOS, NEWSLETTER, etc.). Each category description is a descriptive name defined by the user.

In addition, each category description is preferably associated with a unique identifier (preferably a number, but other identifiers could be used). The identifier is created by the computer (e.g., in sequential order) and used internally to manage the categories. If a user changes the name of a category description, the associated identifier is not changed. However, the invention could be implemented without using identifiers, although changes to category descriptions would then require more maintenance of the FID than with identifiers.

In the preferred embodiment, the identifier is coded so that the initial digit indicates the category type (e.g., "Oxx" indicates the first category type, FILE_TYPE). This aspect is not an essential feature of the implementation of the invention, but helps the program determine, without the need of any other data fields, the column of each category description.

In the preferred embodiment, the user is provided with an FCT containing sample category descriptions. These category descriptions may be changed or deleted, and new categories may be added.

FIG. **3** shows an example of a File Category Table in accordance with the present invention. Category types **1** are at the head of each column, and each column entry comprises a category description **2** and an associated identifier **3**. Of course, in implementing the invention, the table structure shown can be configured in any desired manner, such as an array, linked list, fixed or variable record length table using sequential or hashed access, etc.

In the preferred embodiment implemented under the Apple Macintosh® System **7** operating system, category descriptions are stored as records of a random access data base file. The preferred name of the file is "FC Categories". Records are accessed by record number, in known fashion. In the preferred embodiment, category descriptions are stored in the following data structure:

```
typedef struct FC_CATEG_REC {
  LONG rec_no;                  this record number
  CATEG_SYM sym;                category identifier
  CATEG_NAME name;              category name
  INT list;                     list to which the category
                                belongs (0,1,2,...)
  UINT attr;                    attributes
  UINT valid;                   1 when to be displayed
  LONG file_counter;            how many files use this
                                category
  INT nlinks;                   number of links
  CATEG_SYM alinks[10];         category identifiers of
                                linked categories
  UCHAR filler [128]
} FC_CATEG_REC;
```

During initialization of the file control system, category descriptions (of type FTYP, FDES, CUST, NAME) are read into memory and stored in moveable memory blocks (handles). In the preferred embodiment, data about each category list is gathered in a record structure named COLUMN:

```
typedef struct COLUMN {
  RESTYPE arestype;             resource type of categories
                                in this list (category type)
  INT a_of_nn;                  number of entries in this
                                column-list
  CATEG_RES **ah_to_categs;     handle to array of category
                                descriptions (CATEG_RES
                                structures) in this list
  INT start_cat_vals,
    max_cat_vals;               range of category identifiers
}COLUMN;                        in this list
```

COLUMN entries are stored in an array having the structure:

COLUMN *columns;

File Information Directory Structure

In the preferred embodiment, the FID is a table that can be modeled as having a set of columns labeled by file name, file location (using direct or indirect addressing), creation and/or last update time and date for the file, number of categories associated with the file, and the identifiers of the categories associated with each file by a user.

Other file attributes may be saved in an entry, as desired. For each file that the user may want to locate at a later time, an entry is created in the FID.

FIG. **4** shows an example of a File Information Directory in accordance with the present invention. Each entry has data fields that correspond to the file name (FILE_NAME) **5**, file location (FILE_LOC) **6**, file creation time (DATE/TIME) **7**, number of associated categories (NO_CATEGORIES) **8**, and an array of the identifiers of the associated categories (CATEGORY_ARRAY) **9**.

When the invention is used under some operating systems, the location of each file comprises a record entry in the FID. However, in one embodiment run under the Apple Macintosh® System **7** operating system, the FID entry can be associated with a corresponding standard "Alias Record". An Alias Record contains internal system information about the file, allowing the operating system to find the file quickly even if the file has been moved and/or renamed. More particularly, the data for each file categorized by a user is stored in two resource records. One of the records is called a Catalog Entry and is stored as a resource of type 'Catalog Record'. A Catalog Entry resource record contains information describing the file and its associated category identifiers. The other record is of the type 'Alias Record', and is an Alias Record enabling rapid location of the file in the system by the operating system Alias Manager. Both resource records referring to the same file have the same resource identifiers. During initialization of the file control system, the 'Catalog Record' records are read into memory and stored in an array and serve the common purposes of displaying and selecting categorized files. The associated 'Alias Record' reside on a system storage unit (e.g., magnetic disk drive) and are read into memory only in order to locate and open a file.

Catalog entries are stored in a random access data base file with the preferred name "FC Catalog". Aliases are stored in the random access data base file with the preferred name "FC Aliases".

The preferred embodiment, the structure of the 'Catalog Record' is as follows:

```
typedef struct FC_CRP_DISK {
  LONG rec_no;                       this record number
  UCHAR fnamep [FNAME_L+1]           file name (Pascal
```

-continued

	String) (alas)
UNSIGNED LONG INT credat;	creation time/ date of the file
LONG alias_rec_no	corresponding 'Alias Record' record number
INT ncategs;	number of categories describing the file
CATEG_SYM acategs [MAX_CAT];	array of categories identifiers describing the file
LONG creator;	
LONG type;	
UNSIGNED CHAR filler [56];	
} FC_CRP_DISK;	

CRP records in memory have the same structure. The memory list of CRP records is kept in a moveable block (handle) referred to by:

CRP **hacrp;

Overview of Operation

The invention preferably uses a windows-based graphical user interface for interacting with a user. Such interfaces are common, and include the Macintosh® System 7 operating system from Apple Computer and Microsoft Windows® from Microsoft Corporation.

The invention is activated by running a File Control (FC) Manager program that implements the invention. The FC Manager may be executed ("run") by opening a corresponding file by selecting a menu command and leaving the file open. Other methods known in the art for activating the invention could be used, such as by using a computer mouse to "click" on an icon.

During operation, the FC Manager may be in four states: Inactive, Active, Search Filter Definition, and Categorization. Search Filter Definition is a substate of the Active state. Categorization is generally a substate of the Active state, but in one case, Categorization is a substate of the Inactive state. All states require that the basic data structures described above be initialized. The initialization process initializes all the data structures by allocating memory and reading data from related data files (e.g., the FCT and FID tables, and previously saved "last used" values), in known fashion. After initialization, the FC Manager is set to the Inactive State.

The Active State enables use of the FC Manager to perform such functions as categorizing files, editing the FCT entries, defining a search filter, and locating and opening files.

1. Categorizing Files

Categorization of files in the illustrated embodiment is performed by the user from a Categories Window. In the illustrated embodiment, there are two categorization states—Active and Inactive. In the Active state, a user need only close an uncategorized file to immediately be presented by the FC Manager with the Categories Window. Typically, from within an application, the user will open a file (or, having created a new file, will make the first save to disk), at which time the FC system extension, running as a background process, will detect that action and store the path to the file in common memory. The FC Manager, running as a concurrent process, during "null events" (i.e., periods of inactivity) will retrieve this path from common memory and check the path against a list of already categorized files. If the file has not yet been categorized, the FC Manager will automatically categorize the file with the special category "Uncategorized", and notify the user that there are files to be categorized.

In the Inactive state, to categorize an open file, the user must affirmatively choose a command from a menu, e.g., a "Categorize" command, to run the FC Manager. When in the Inactive state, the user may open a file specifically for the purpose of selecting categories for that file.

In the preferred embodiment, when the FC Manager is running as a background process, each file being closed is submitted to conditional categorization, which means that only uncategorized files undergo the operation. The file being closed is identified and checked to determine whether it already has an entry in the FID. The file is assumed to be already categorized if there is an entry in the FID with the same creation date and path. In this case, no further action is taken. Otherwise, the FC Manager opens the Categories Window and the user is asked to categorize the file.

When the FC Manager opens the Categories Window, the category names stored in the FCT are displayed, Preferably, the Categories Window presents category descriptions in an organized fashion. It has been found that organizing the category names into columns helps the user to locate and select desired categories. In the illustrated embodiment, the category descriptions are displayed alphabetically in columns, with each column having a heading comprising the category type. Preferably, all category descriptions within a heading are related. However, it is often unclear in which column a given category belongs. Accordingly, the column position of a category is not significant. Columns are used for the convenience of the user in finding relevant categories and for no other reason. Further, any desired display of the category descriptions can be used.

FIG. **5** shows an example of a file manager display in accordance with the present invention. A Categories Window **50** is shown on the right side of the display, with a File Window **52** on the left side. Category types **54** are shown at the top of the Categories Window **50**, with category descriptions **56** arrayed in columns below the category types **54**. A tally **58** is displayed of the number of files matching selected categories, as further described below.

While the Categories Window **50** is displayed, both the category type headings and the column positions of the category descriptions may be edited by the user. In addition, category types may be added or deleted. Further, category descriptions may be edited or deleted, and new category descriptions may be added while in the Categories Window. A user is preferably warned before deleting a category description or category type. After user confirmation of deletion, the category descriptions is removed from the FCT, and all references in the FID entries to that category description are also removed. Implementation of such editing functions is known in the art of data processing, and simply cause the associated FCT and FID records to be updated.

A category description may be moved from one column (category type) to another column. In the preferred embodiment, moving is accomplished by clicking on the category description with the mouse and dragging and dropping the category description in the new column. In the illustrated embodiment, moving a category description between category types changes the category type numeric value, so all associated records in the FID containing the moved category description are checked and modified.

Categories which describe the current file can be selected by the user. This is done, for example, by pointing a computer mouse and "clicking" on each category description to be applied to the file. Preferably, the user may select as many category descriptions as apply to the particular file. After the user has completed category selection for a file, a new entry in the FID is created using the file data associated with the file, and all of the selected category descriptions. When the user has completed category selection, the Categories Window may be closed by the user, or the user may select another file for categorization.

In the preferred embodiment, a selected file that has been categorized may be recategorized using the FC Manager by clicking a "Categorize" button **60**. This does not require any further file identification, as the file selected by the user contains the information necessary to perform categorization.

If the user adds more category descriptions while the Categories Window is displayed, the FC Manager also creates appropriate new unique identifier entries and the corresponding category description entries in the FCT. In subsequent references to the category descriptions by the FC Manager, these identifiers are used, the corresponding names of the categories being the way the user "connects" with these identifiers. This means that if the user changes the name of a category description, the associated identifier is not changed, thus maintaining the validity of all prior uses of that identifier in the FID.

As an example, assume a file named "jones.mem" located in the "c:\memos" subdirectory and having the file date/time of "01-01-80 01:30" is categorized by a user under the two category descriptions "MEMOS" and "JONES" (see FIG. **3**). The FC Manager would make an entry in the FID table (see FIG. **4**) as follows (field names are supplied for convenience):

[FILE_NAME	FILE_LOC	DATE/TIME	NO_CAT.	CAT_ARRAY]
jones.mem	c:\memos	01-01-80 01:30	2	008, 300

The two identifiers "008" and "300" are obtained from the FCT as the identifiers associated with the category descriptions "MEMOS" and "JONES", respectively. If the category description "MEMOS" is later changed to "NOTES", then the FID entry for "jones.mem" would still be linked to the category description "NOTES" by the identifier "008".

In the preferred embodiment, when the user clicks a "Show Usage" command from the Categories Window menu, the FC Manager checks the FID for the number of references to each category description and displays these numbers next to each category description. This feature enables the user to see which category descriptions are frequently used and which are not used at all. The unused or little used category descriptions may then be deleted, and others may be moved around for better accessibility.

If desired, the process of categorizing existing files can be totally or partially automated. After a number of files have been categorized, word patterns in categorized files can be correlated to the category descriptions. This information can be used to automatically assign (or simply suggest) category descriptions to new and existing uncategorized files.

Further, when categorizing a more extensive, broadly based set of files, category descriptions can be grouped into (overlapping) subjects or projects, with a short list of subjects or projects (effectively a top level category set) shown in a separate window (or on a menu). Once a subject or project is chosen, the category descriptions list is shortened to only show those relevant to that subject or project. Taking that idea further, the inventive categorization system can be used recursively. For example, if all the topics in the Library of Congress were categorized, the number of category descriptions needed would be impracticably large and unmanageable. If that list of category descriptions were

itself regarded as the data, the invention can be applied to manage a "higher level" category list to manage and access limited portions of the complete category description list. This kind of "multi-level" categorization can be carried to any depth needed. In essence, this approach is another way of organizing lists of category descriptions of the type contemplated by the present invention, adding back some flavor of a hierarchical structure but with the added benefits of precision of input and certainty of existence.

In summary, categorization of a file in accordance with the invention involves the following steps:

1. defining a list of category descriptions;
2. associating one or more category descriptions with a file; and
3. storing a file record containing file identity information, file location information, and the associated category description(s) for the file.

Finding Files

In the process of search and retrieval, the invention overcomes the problem of search filter definition, ensuring that the user defines a filter which will always find at least one file, thus avoiding wasting time in searching for data that cannot be matched. This is achieved in two ways. First, the user is not required to type the key words to search; instead, the user simply chooses the words in random order from pick lists, making mistyping impossible. Second, as the user builds the search filter definition, categories which would find no data are automatically excluded as pick list possibilities.

In order for a user to find one or more files, the Search Filter Definition state is invoked by opening the Categories Window to display the existing category descriptions (see FIG. **5**). The user initiates definition of a search filter by, for example, clicking the mouse on a "Set Categories" button. The user then selects the pre-defined category descriptions for the files which the user wants to find. In the illustrated embodiment, this is done in the same way as when categorizing a file: the user clicks the mouse on each applicable category description. After each click, the categories listed in the Categories Window are narrowed to show only those other categories which are used with the selected categories for at least one other file. The category descriptions may be selected in any order. Moreover, those category descriptions whose selection would not change the matching file list are disabled (e.g., shown as "grayed"), as further described below.

The selected categories comprise the user's search filter, which is said to define, or "map", a hybrid folder. The search filter is used to search through the FID table entries for files which match the search filter criteria, and hence come within the specified hybrid folder (that is, the search locates those file records in the FID that have identifiers that match the identifiers of the selected categories).

Any search technique may be used to search the FID table entries. For example, the FID entries may be sequentially searched and each identifier value in the CATEGORY_ARRAY field for each entry compared to the identifiers of the category descriptions selected by the user. In an alternative embodiment, a concordance file indexed by identifier value may be constructed from the FID entries, and a search conducted through the concordance file to generate a list of FID entries matching all search filter identifier values.

In addition to selecting category descriptions for the search filter, the user preferably may also group the categories, and relate the groups with logical connectors. Of course, a group may include just one category description. Although current implementations provide for an automatic "AND" conjunction between selected categories, other logical operators may be used, such as an explicit "AND" operator, or the "OR" or "NOT" operators. For example, the user may define a search filter of "MEMOS AND JONES". The FID would be searched for all entries having both the identifiers "008" (for "MEMOS") and "300" (for "JONES"). In the example shown in FIG. 4, this search filter would find at least the document "jones.mem". Although other searching mechanisms allow similar searches, none use methods which ensure certainty of existence.

In the preferred embodiment, the Categories Window display indicates how many files match the present search filter. In the illustrated embodiment, the actual number of files in the hybrid folder is displayed. However, the file names in the hybrid folder could also be displayed for this purpose.

It is expected that the FID will be relatively small for most implementations, which allows for very quick searching. In defining a search filter, the hybrid folder to which it maps should contain a sufficiently small number of files to make accessing a particular file easy. Thus, a user would normally continue to select category descriptions until the defined hybrid folder contains no more than a few (e.g., 10) files.

Once FID entries matching the search filter are located, the corresponding file names in the hybrid folder are preferably displayed in a File Window **52** (see FIG. **5**). The user may then select one or more of the displayed file names, which causes the corresponding files to be retrieved from the appropriated storage device and opened (if only one file is in the hybrid folder, selecting the file can be automatic, thereby obviating display of the file name and manual selection). In operation, the selected file names together with their locations from the selected FID entries are passed to the operating system, which opens the files. Because the FID contains the location of all categorized files on disk, it is not necessary to search any other part of the disk, an action which could be very time consuming.

More particularly, in the preferred embodiment, the usual method of opening a file through the FC Manager is double-clicking on the file entry in the File Window. An alternate method is by selecting a file (by clicking on its entry or typing the first few letters of its name) and clicking on an "Open" button. Both methods can allow the opening of multiple files by using multiple selections, as is known in the art. In the preferred embodiment, the opening process consists of the following steps:

1. The selected entry in the File Window points to the corresponding FID entry. Using this pointer, the associated alias record is retrieved.
2. The alias record is passed to the operating system Alias Manager with the order for "Vast resolution". This method is able to find the file very quickly even if it has been renamed and/or moved to another system folder. If the file name has changed, the user is asked to confirm the new name.
3. If a file has been moved (not copied but moved) to another volume after categorization, and the system has been restarted, then one of the system identifiers (the File ID, which unique is within each volume) for the file has been lost. In such cases, the FC manager performs an exhaustive search of all files on all mounted volumes, searching only for files with the identical creation date and time of the subject file. The user is notified of this action.
4. The result of both searches may be a list of possible matches rather than just a single match (for example, if the file was duplicated and renamed). The FC Manager searches this list looking for files with the same name and creation date/time as in the FID entry. If none are found, then the user is presented with a list of matches with full paths found by the Alias Manager and is asked to select one for opening.
5. When a file is selected, with or without the help of the user, the alias record is checked and updated if necessary to account for any change in file name and/or location.

The result of the search process is a set of standard file identifiers, that is file name, volume reference number, and directory identification. These are passed to the opening routines in the operating system.

In the preferred embodiment, a user is prevented from defining an empty hybrid folder. All category descriptions are disabled which, if added to the search filter defined by the user, would result in no matching files. Disabling of category descriptions may by shown in many ways. For example, disabled category descriptions may be given different display attributes, such as being grayed or dimmed. Alternatively, the names of disabled category descriptions may simply not be displayed. As another alternative, the user may inhibit disabling category description list contraction by selecting a "Full Lists" option.

Determining which category descriptions are to be disabled may be accomplished in several ways. For example, in one embodiment, when no categories are selected, those categories whose identifiers are not used with any files are disabled. When one or more enabled categories are selected, the FID is searched for all files which have

FID entries using all of the identifiers of all of the categories presently selected, and a determination is made as to which other categories are also used on those files. These other categories remain enabled for further selection, but all other categories are disabled.

For example, if a search filter initially includes only the category description "MEMOS", the FC Manager searches the FID for entries containing the identifier corresponding to "MEMOS". Suppose the FC Manager finds three matching files, one being categorized with the categories "MEMOS", "URGENT", and "OFFICE", another being categorized with the categories "MEMOS", "SENT", and "E-MAIL", and the third being categorized with the categories "MEMOS", "SENT", and "LETTERS". The FC Manager creates a union of the set of identifiers for all of the categories found. In this example, the union is "MEMOS", "URGENT", "OFFICE", "SENT", "E-MAIL", and "LETTERS". All categories not in this union are then disabled.

Stated another way, the FID entries are searched for matches to the most recently selected category description in a first step. As a second step, the category description identifiers in the matching FID entries are used to determine which of the remaining category descriptions are not linked by their identifiers to such entries. For example, if a search filter definition initially includes the category description "MEMOS", the FC Manager searches the FID for entries containing the identifier corresponding to "MEMOS". If, for instance, ten entries were located containing the "MEMOS" identifier, then the FC Manager uses the other identifiers in those ten entries to determine which category descriptions to disable. For instance, if none of the ten entries contain identifiers for the category descriptions "SALES" or "PATENT", then those two category descriptions would be disabled. The preferred embodiment for disabling unselectable category descriptions has the effect of making disabling cumulative, so that as the user combines more and more category descriptions in the search filter definition, more and more category descriptions will be disabled. The user chooses from an increasingly limited selection of category descriptions, but will always be assured of having at least one file in the defined hybrid folder.

The ability of the invention to suppress or disable category descriptions that will not result in a match allows definition of a logical operation called "AND PERHAPS". This operation represents a conditional "AND", which means the "AND" conjunctive operation unless with the added search term there would be no match, in which case the term is omitted from the search filter. For example, if the search filter is "MEMOS AND PERHAPS JONES", and "MEMOS" alone would have at least one match, but the addition of "JONES" would cause a non-match, then the "JONES" term is automatically excluded, with a message to the user.

Preferably, the user may also include in the search filter a range of file creation dates and/or times. Preferably, the user may select a predefined range, such as "Last 30 Days". In the illustrated embodiment, the default range is "All Dates". Preferably, those predefined date ranges which would result in no matched files if selected would be disabled. It should be understood that, in general computer systems store file creation date/time as a single relative creation time. Therefore, definition of the date range in the search filter would require a date-to-relative time conversion, as is known in the art. In alternative embodiments, other search criteria may be applied, such as file type, creator type, date last modified, or date last accessed.

To summarize, when a user clicks on a previously unselected category description, that category description is highlighted, the category description is added to the search filter definition, the current number of matching files entered in the FID is computed and displayed, each of the remaining category descriptions are evaluated to be displayed or disabled, and the new list of category descriptions is displayed in the Category Window. Generally, as each category description is selected, the list of category descriptions shrinks because of the removal of all category descriptions which, if checked further, would give zero matching files. In addition, the number of files matching the specified category descriptions is immediately shown in the File Window. These actions prevent the user from defining a hybrid folder which contains no files.

Deselecting a previously selected category description invokes the same routines, except that the result is generally an increase in the number of listed category descriptions. Widening or narrowing the date filter has a similar effect on the displayed list of category descriptions.

Preferably, the last search filter and hybrid folder are stored on a storage medium for future use. (Alternatively, the last n, or all, prior search filters and hybrid folders may be stored for future use). Thus, when a user desires to access a file, the user is immediately presented with the hybrid folder as last defined. In many instances, the desired file will be in the last hybrid folder created. To maintain information about the Categories Window and File Window, it is useful to store certain data. In the preferred embodiment, such catalog data is stored in the following record structure:

typedef struct SelectedFiles {	
INT file_match;	number of files matching the current search filter (i.e., the number of files to be displayed in the File Window)
INT *a_of_match_inds;	pointer to an array containing indexes to FID entries of files matching the current search filter
INT file_marked;	number of files selected by the user in the File Window
INT *a_of_marked;	pointer to an array of indexes to FID entries for the files selected by user
INT *a_of_rs;	pointer to an array of row numbers in the File Window for selected files (i.e., their positions relative to the window)
INT from_ind;	starting index for file search in the FID catalog
INT buttons_on;	indicates whether special File Window buttons are enabled
} SelectedFiles;	

All three arrays pointed to by the structure members are dynamic, meaning that their size changes according to current needs.

In summary, finding a file in accordance with the invention involves the following steps:

1. defining a search filter of category descriptions from a pre-defined list of category descriptions;
2. searching the category descriptions of each previously stored file record for a logical match to the category descriptions of the defined search filter;
3. optionally, disabling selection of all category descriptions that would not provide a match to the defined search filter; and
4. displaying at least part of the file identity information of all records having category descriptions that logically match the category descriptions of the defined search filter.

Outdated FID

Various events may impact the integrity of the identifiers in the FID, such as changes to any part of the fully qualified file path (volume or drive, directory chain, and file name). Such events include:

1. Moving the file into another directory.
2. Changing the file name.
3. Renaming the directory containing the file.
4. Moving the directory containing the file into a different directory.
5. Moving the file to another volume (disk).
6. Deleting the file.

When the invention is implemented under the Apple Macintosh® System 7 operating system utilizing Alias Records, events 1, 2, 3, and 4 have no impact because the file ID (used to access files) is part of the Alias Record and is not changed unless the file is moved to another volume. Preferably, if the user attempts to open a file for which the name was changed, the user will be notified of that fact and will be asked to confirm that the file is the correct one.

Events 5 and 6 may cause a delay before the file is found (for event 5) or determined to no longer exist (in which case, an error message is returned). In the illustrated embodiment, when either event 5 or 6 occurs, the Alias Record for the file is updated (after user confirmation) in the FID. In this way, the FID is kept current after each access or attempted access.

On systems that do not provide Alias Records, the FID may store the fully qualified file path (volume or drive, directory chain, and file name) for the file and the file creation date/time. In such a case, a means is preferably provided for finding files which are not at the location indicated in the corresponding FID entry. For example, a search may be made, beginning with the directory closest to the file's original location, using the creation date/time as the search criteria. It has been found that a file with the identical creation date/time (to the nearest second) of the searched file is generally the desired file. However, there may be situations (such as duplicate files) where more than one such file is found. In that case, the file names of the found files are presented to the user, who selects one.

It should be noted that the present invention may be embodied as a replacement for an operating system, thus replacing the hierarchical file structure, or allowing both the hierarchical and the hybrid file structures to be used together. In such an embodiment, features for resolving ambiguities and keeping track of moved or renamed files can be further optimized using the low level access to the disk directories which the operating system controls.

Special Categories

In the preferred embodiment, category descriptions may be further characterized as "standard" or "special". The characterization of a category description may be indicated by setting a flag field in the corresponding FCT record, as provided in the data structure CATEG_RES described above.

The standard category has the attributes as described above. One special category is the "linking" category. A linking category description is linked to other "linked" category descriptions. A linking category provides for the situation when a file is described by one category description, and the file should also be described by a related category description. Such linkage may be indicated in the corresponding CATEG RES data structure described above, by recording an array of identifiers for linked category descriptions. A category may be both a linking category and a linked category.

As an example of linking and linked categories, a category named "E-Mail" could be defined as a linking category and linked to the category descriptions "Sent", "Received", "Action", "Urgent", and "Reply". Thus, when a file is categorized as "E-Mail", the user would be given an indication (by any convenient method) that the file should also be assigned to one of the linked category descriptions "Sent", "Received", "Action", "Urgent", or "Reply".

Preferably, when the user selects a linking category, the user is reminded to also select from the corresponding linked category descriptions. The linked categories may be indicated in the Categories Window using a distinctive style such as underline or bold, or using a check mark. Alternatively, a dialogue box may be displayed containing the linked category descriptions. If the user fails to select at least one of the linked category descriptions, a warning dialogue is preferably displayed.

If a user desires to delete a linking category, the linked categories are preferably indicated. The user is preferably given the opportunity to select linked categories to delete.

Another special category type can be called "encrypted". An encrypted category requires a password which is used to initiate encryption of all files which have that category. Encrypted files may not be accessed except by first entering the necessary passwords. For example, the user would be asked to enter the password for each such category description selected when creating a search filter. Since the file contents are encrypted, access without the necessary passwords would only reveal unintelligible codes.

Since files may be assigned to multiple category descriptions, more than one password may be required to access a file. However, once each password is entered, all files using that password are made accessible through the FC Manager. This method of encrypting files has an advantage over current methods in that once a password is entered, the user may access any number of files without further needing a password. Current methods of protecting individual files usually require entry of a password each time the file needs to be opened.

Many other implementations of these ideas are possible in electronic network information management, electronic mail, or any other data management systems. For example, in a large office or on a public or private electronic network, communication between people can be difficult because of the very large number of people trying to communicate. After a certain point, there is an information overload and it is too time consuming to try to search through all the messages for the few of interest. Using the invention, a category description list could be defined for all possible topics (with constant updating by the network administrator and/or by the users). Each user could then use this list of category descriptions to both post messages and search for messages on topics of interest to the user. The user could also set up sets of search filters for particularly important topics. If any messages were to be posted whose topics matched those search filters, they would be immediately sent to the user.

Accordingly, the present invention provides a method for accessing files which has intuitive access by user-defined topics. More particularly, the invention provides: easy access to a large number of files and to files having overlapping categories; simple access to files stored in a hierarchical file system without the necessity of sorting through multiple levels; access to files using predefined categories descriptive of the contents of the files; access to files which permits a user to create a search filter of categories of files using precise category names to which the files belong, with the assurance that the filter will always find some files (although possibly deleted); and a method of accessing files which is unaffected by changes in file contents.

A number of embodiments of the present invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. For example, although the above description has been made with respect to a single computer system, that term is meant to include distributed data storage environments, such as networked computers. Further, although the above description has contemplated that the "user" is a person, the invention can be readily adapted to interact with another computer as the "user". Accordingly, it is to be understood that the invention is not to be limited by the specific illustrated embodiment, but only by the scope of the appended claims.

We claim:

1. A method for accessing files in a data storage system of a computer system having means for reading and writing data from the data storage system, displaying information, and accepting user input, the method comprising the steps of:

(a) initially creating in the computer system a category description table containing a plurality of category descriptions, each category description comprising a descriptive name, the category descriptions having no predefined hierarchical relationship with such list or each other;

(b) thereafter creating in the computer system a file information directory comprising at least one entry corresponding to a file on the data storage system, each entry comprising at least a unique file identifier for the corresponding file, and a set of category descriptions selected from the category description table; and

(c) thereafter creating in the computer system a search filter comprising a set of category descriptions, wherein for each category description in the search filter there is guaranteed to be at least one entry in the file information directory having a set of category descriptions matching the set of category descriptions of the search filter.

2. A method for accessing files in accordance with claim **1**, wherein each category description comprises a user defined category name and a unique category description identifier created by the computer system.

3. A method for accessing files in accordance with claim **2**, wherein each category description further comprises a category type designation.

4. A method for accessing files in accordance with claim **2**, wherein the step of creating a category description table comprises the steps of:

(1) accepting user input defining a new category description;

(2) displaying the new category description;

(3) creating a unique category description identifier associated with the new category description; and

(4) storing the new category description and unique category description identifier in the category description table.

5. A method for accessing files in accordance with claim **4**, wherein the step of creating a file information directory comprises the steps of:

(1) accepting user input selecting a file;

(2) displaying each category description in the category description table;

(3) accepting user input associating the selected file with at least one category description selected by the user from the displayed category descriptions;

(4) creating a new entry in the file information directory;

(5) storing in the new entry the file identifier of the selected file; and

(6) storing in the new entry the category description identifier of each of the selected category descriptions.

6. A method for accessing files in accordance with claim **5**, wherein the data storage system further includes a file name for each file and the step of selecting a file comprises the steps of:

(1) displaying at least one file name; and

(2) accepting a user selected of a file from the displayed file names.

7. A method for accessing files in accordance with claim **1**, wherein the step of creating a search filter comprises the steps of:

(1) disabling category descriptions which if added to the search filter would not match the category descriptions of at least one entry in the file information directory;

(2) accepting user input selecting at least one category description as a component of the search filter.

8. A method for accessing files in accordance with claim **7**, wherein disabled category descriptions are indicated by means of a unique display attribute.

9. A method for accessing files in accordance with claim **7**, wherein the step of creating a search filter further comprises the steps of;

(1) relating at least two selected category descriptions with logical operations.

10. A method for accessing files in accordance with claim **7**, wherein the data storage system includes the time of creation of each file, and the step of creating a search filter further comprises the step of:

(1) accepting user input defining a time range to limit matching to only those entries in the file information directory having creation times in the selected time range.

11. A method for accessing files in accordance with claim **7**, wherein each category description comprises a user defined category description name and a unique category description identifier created by the computer system, and further including the step of displaying the name of each file in the file information directory having category description identifiers matching the category description identifiers of the category descriptions in the search filter.

12. A method for accessing files in accordance with claim **11**, further comprising the steps of:

(1) selecting one of the displayed file names; and

(2) opening the file corresponding to the selected file name.

13. A method for accessing files in accordance with claim **12**, wherein the step of selecting a file name comprises the step of:

(1) accepting user input selecting a file from the displayed file names.

14. A method for accessing files in accordance with claim **12** wherein the step of opening a file comprises the step of:

(1) testing if only one file name is displayed, and if so, then opening the file corresponding to that file name.

15. A system for accessing files in a data storage system comprising:

(a) a plurality of files in a data storage system;

(b) a plurality of user-defined category descriptions, each category description comprising a descriptive name, the category descriptions having no predefined hierarchical relationship with such list or each other;

(c) file association means for associating at least one file with at least one category description selected from the plurality of previously defined category descriptions;

(d) category description addition means for adding one or more additional category descriptions to the plurality of user-defined category descriptions; and

(e) category linking means for linking at least one linking category description to at least one linked category description, such that if a specific file is associated with a linking category description, the user must also associate that specific file with at least one of the linked category descriptions corresponding to the linking category description.

16. A system for accessing files in a data storage system as set forth in claim **15**, further including a linking reminder activated upon associating a specific file with a linking category description, such that the linking reminder provides an indication to the user that the user must associate that file with at least one of the linked category descriptions corresponding to the linking category description.

17. A system for accessing files in a data storage system as set forth in claim **15**, further including password-controlled file encryption means for encrypting at least one of the plurality of files, such that if a specific file is encrypted, associating that file with a category requires provision by the user of a password, and any access to that file requires provision by the user of a password.

18. A system for accessing files in a data storage system of a computer system having means for reading and writing data from the data storage system, displaying information, and accepting user input, wherein each file located on the data storage system has a file name, the system comprising:

(a) means for initially defining in the computer system at least one list having a plurality of category descriptions, each category description comprising a descriptive name, the category descriptions having no predefined hierarchical relationship with such list or each other;

(b) means for thereafter accepting user input associating with a file at least one category description from at least one defined list;

(c) means for storing in the data storage system a file record containing at least the file name, file location information, and the associated category descriptions for the file;

(d) means for displaying from each defined list, as selectable items, only those category descriptions associated with at least one file;

(e) means for accepting user positional input defining a search filter of at least one category description selected from at least one displayed defined list;

(f) means for automatically disabling, in the computer system, selectability of all other category descriptions in each displayed list that do not have associated files which are also associated with the category descriptions of the defined search filter;

(g) means for searching in the computer system the category descriptions of each stored file record for a logical match to the category descriptions of the defined search filter;

(h) means for displaying the file names of all file records having category descriptions that logically match each category description of the defined search filter.

19. The system for accessing files of claim **18**, further including:

(a) means for accepting user input selecting at least one file from the displayed file names;

(b) means for accessing each selected file on the data storage system using the file location information from the file record associated with each corresponding selected file.

20. A method for accessing files in a data storage system of a computer system having means for reading and writing data from the data storage system, displaying information, and accepting user input, wherein each file located on the data storage system has a file name, the method comprising the steps of:

(a) initially defining in the computer system at least one list having a plurality of category descriptions, each category description comprising a descriptive name, the category descriptions having no predefined hierarchical relationship with such list or each other;

(b) thereafter accepting user input associating with a file at least one category description from at least one defined list;

(c) storing in the data storage system a file record containing at least the file name, file location information, and the associated category descriptions for the file;

(d) displaying from each defined list, as selectable items, only those category descriptions associated with at least one file;

(e) accepting user positional input defining a search filter of at least one category description selected from at least one displayed defined list;

(f) automatically disabling, in the computer system, selectability of all other category descriptions in each displayed list that do not have associated files which are also associated with the category descriptions of the defined search filter;

(g) searching in the computer system the category descriptions of each stored file record for a logical match to the category descriptions of the defined search filter;

(h) displaying the file names of all file records having category descriptions that logically match each category description of the defined search filter.

21. The method for accessing files of claim **20**, further including the steps of:

(a) accepting user input selecting at least one file from the displayed file names;

(b) accessing each selected file on the data storage system using the file location information from the file record associated with each corresponding selected file.

* * * * *

Exhibit 11

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

```
                              )
PARTSRIVER, INC.,             )
   Plaintiff,,                )
                              )
v.                            )
                              ) CASE NO. 2-07-CV-440 DF
SHOPZILLA, INC., YAHOO!       )
INC.; PRICEGRABBER.COM,       )
INC.; EBAY, INC.; and         )
MICROSOFT CORPORATION,        )
   Defendants.                )
```

****

ORAL AND VIDEOTAPED DEPOSITION OF

SHERIF DANISH

VOLUME 1

JANUARY 20, 2009

****

ORAL AND VIDEOTAPED DEPOSITION OF SHERIF DANISH, produced as a witness at the instance of the DEFENDANTS, and duly sworn, was taken in the above-styled and numbered cause on the 20th day of January, 2009, from 9:07 a.m. to 5:02 p.m., before Julie C. Brandt, RMR, CRR, and CSR in and for the State of Texas, reported by machine shorthand, at the offices of Fulbright & Jaworski, 2200 Ross Avenue, Suite 2800, Dallas, Texas, pursuant to the Federal Rules of Civil Procedure and the provisions stated on the record or attached hereto.

SHERIF DANISH,

having been first duly sworn, testified as follows:

EXAMINATION

BY MR. ZEMBEK:

Q. Good morning.

A. Good morning.

Q. Could you please state your full name for the record.

A. Sherif Danish.

Q. And where do you live, Mr. Danish.

A. Cupertino, California.

(Exhibit 1015 marked.)

Q. I'm going to hand you Deposition Exhibit 1015. Do you recognize this?

A. It's a response to interrogatories, yeah.

Q. Okay. Have you seen these before?

A. I remember I received a document that had questions to me, to which I answered, like possibly the documents that I sent, but I don't recall this specific document.

Q. Could you go to the next to last page? There's a declaration.

A. Yeah.

Q. Do you remember signing that?

A. Yeah.

Q. Did you review the interrogatory before you signed that?

A. Sir, it was one of the documents to which I was asked -- on which I was asked questions.

Q. Who asked you the questions?

A. In the document itself. Right here, identify all assignments, licenses, sublicenses --

Q. What did you do to answer those questions?

A. To the best of my knowledge, I just need to look at it again. I mean, I don't have --

Q. Take your time.

A. -- enough time. Like, for example, question number 1.

Q. Yes.

A. Describe in detail the facts and circumstances of the conception of the alleged invention. So I don't remember exactly how this happened, but I did describe -- I don't know if it was by phone or in a meeting or whatever, the facts of -- and the circumstances of the conception of the patent.

Q. Okay.

A. Like it's stated in the response here.

0d6d4bf6-b223-4946-869a-e5bdfa2117a0

Q. Okay. And what is a server?

A. In the internet environment, that would be a web server, for example.

Q. And what is a client?

A. The client on the internet would be a web browser.

Q. How do you accomplish displaying the feature screen indicating the alternatives in your patent?

A. How do I --

Q. How is that accomplished? How do you do it?

A. You send to the browser -- in the case of the internet --

Q. Uh-huh.

A. -- you would send to the browser of the client the list of attributes and the values of each attribute for that particular product family, for example, like printers, like we were saying.

Q. And how do you send it to the browser? What do you send to the browser?

A. There is a protocol, I guess it was described here, an HTTP protocol where the browser requests information from the server, like show me the printer catalog, for example. And the server in response to the HTTP request will send the attributes and attribute values.

Q. And what does the browser do once it gets the attributes and the attribute values?

A. It displays them.

Q. Does the browser process HTML code that it receives from a server?

A. Yes.

Q. What does it do when it processes the HTML code?

A. It formats the text.

Q. Okay. And after it formats the text, what does it do?

A. Puts it on the screen.

Q. What do you mean it "puts it on the screen"?

A. Displays it on the screen.

Q. Is there any teaching of displaying on the server in column 18 or 19?

A. Displaying on the server?

Q. Yes.

A. The display, if I recall, is a display that happens on the user of the -- on the browser of the client.

Q. Okay. And when did you first start working on the internet implementation?

A. Beginning of the summer of '94. I would say approximately around that time.

Q. And when did you complete a prototype of the internet implementation?

A. By the end of the summer of '94.

Q. And does that prototype still exist?

A. I don't think so, no.

Q. The prototype that you developed in the summer of '94, was that a client server implementation?

A. It was an internet implementation with a browser and server, web server.

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

```
                                   )
PARTSRIVER, INC.,                  )
   Plaintiff,,                     )
                                   )
v.                                 )
                                   ) CASE NO. 2-07-CV-440 DF
SHOPZILLA, INC., YAHOO!            )
INC.; PRICEGRABBER.COM,            )
INC.; EBAY, INC.; and              )
MICROSOFT CORPORATION,             )
   Defendants.                     )
```

REPORTER'S CERTIFICATION
DEPOSITION OF SHERIF DANISH, VOLUME 1
JANUARY 20, 2009

I, Julie C. Brandt, Certified Shorthand Reporter in and for the State of Texas, hereby certify to the following:

That the witness, SHERIF DANISH, was duly sworn by the officer and that the transcript of the oral deposition is a true record of the testimony given by the witness;

That the deposition transcript was submitted on ________________________ to the witness or to the attorney for the witness for examination, signature and return to Merrill Legal Solutions by ________________________;

That the amount of time used by each party at the deposition is as follows:
MR. ZEMBEK.....05 HOUR(S):40 MINUTE(S)
MR. CEDEROTH.....00 HOUR(S):00 MINUTE(S)
MR. TOMASULO.....00 HOUR(S):00 MINUTE(S)
MS. DeRIEUX.....00 HOUR(S):00 MINUTE(S)
MR. COYKENDALL.....00 HOUR(S):00 MINUTE(S)

That pursuant to information given to the deposition officer at the time said testimony was taken, the following includes counsel for all parties of record:

FOR THE PLAINTIFF:

Robert W. Coykendall
MORRIS LAING
300 N. Mead
Suite 200
Wichita, Kansas 67202-2722
316.262.2671
316.262.6226 (fax)
rcoykendall@morrislaing.com

Jack Baldwin
BALDWIN & BALDWIN, LLP
400 West Houston Street
Marshall, Texas 75670
903.935.4131
903.935.1397 (fax)
jbb@baldwinlaw.com

FOR THE DEFENDANTS MICROSOFT and eBAY:

Richard A. Cederoth
SIDLEY AUSTIN, LLP
One South Dearborn
Chicago, Illinois 60603
312.853.7026

312.853.7036 (fax)
rcederoth@sidley.com

Theodore W. Chandler
SIDLEY AUSTIN, LLP
555 West Fifth Street
Suite 4000
Los Angeles, California 90013
213.896.5830
213.896.6600 (fax)
tchandler@sidley.com

FOR THE DEFENDANT PRICEGRABBER.COM:

Michael A. Tomasulo
JONES DAY
555 South Flower Street
Fiftieth Floor
Los Angeles, California 90071
213.243.2619
213.243.2539 (fax)
matomasulo@jonesday.com

FOR THE DEFENDANT SHOPZILLA, INC.:

Elizabeth L. DeRieux
CAPSHAW DeRIEUX, LLP
1127 Judson Road
Suite 220
Longview, Texas 75601
903.236.9800
903.236.8787 (fax)
ederieux@capshawlaw.com

FOR THE DEFENDANT YAHOO! INC.:

Richard Zembek
FULBRIGHT & JAWORSKI
1301 McKinney
Suite 5100
Houston, Texas 77010-3095
713.651.5203
713.651.5246 (fax)
rzembek@fulbright.com

Dan D. Davison
FULBRIGHT & JAWORSKI
2200 Ross Avenue
Suite 2800
Dallas, Texas 75201-2784
214.855.8049
214.855.8200 (fax)
ddavison@fulbright.com

That $__________ is the deposition officer's charges to the Defendant Yahoo! Inc. for preparing the original deposition transcript and any copies of exhibits;

I further certify that I am neither counsel for, related to, nor employed by any of the parties or attorneys in the action in which this proceeding was taken, and further that I am not financially or otherwise interested in the outcome of the action.

Certified to by me January 29th, 2009.

Julie Brandt
Julie C. Brandt, CSR, CRR
Texas CSR No. 4018
Expiration Date: 12/31/10

Merrill Legal Solutions
Reg. No. 191
4144 North Central Expressway
Suite 850
Dallas, Texas 75204
800-966-4567

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

```
                              )
PARTSRIVER, INC.,             )
   Plaintiff,,                )
                              )
v.                            )
                              ) CASE NO. 2-07-CV-440 DF
SHOPZILLA, INC., YAHOO!       )
INC.; PRICEGRABBER.COM,       )
INC.; EBAY, INC.; and         )
MICROSOFT CORPORATION,        )
   Defendants.                )
```

****

ORAL AND VIDEOTAPED DEPOSITION OF

SHERIF DANISH

VOLUME 2

JANUARY 21, 2009

****

ORAL AND VIDEOTAPED DEPOSITION OF SHERIF DANISH, produced as a witness at the instance of the DEFENDANTS, and duly sworn, was taken in the above-styled and numbered cause on the 21st day of January, 2009, from 9:39 a.m. to 4:49 p.m., before Julie C. Brandt, RMR, CRR, and CSR in and for the State of Texas, reported by machine shorthand, at the offices of Fulbright & Jaworski, 2200 Ross Avenue, Suite 2800, Dallas, Texas, pursuant to the Federal Rules of Civil Procedure and the provisions stated on the record or attached hereto.

Q. Let me go back and make the whole question clear.

Under my example where the user is in Texas and the web server is in California, are any of the steps 1 A through 1 K, are any of those steps performed entirely in Texas, in your opinion?

MR. BALDWIN: Object to the form.

A. Well, if you look at step G, revising said feature screen, "revising" here obviously means displaying, again. So it is equivalent to the analysis of step C that I just did.

Q. What does the term "revising said feature screen to indicate the available alternatives of the first subfamily," what does that mean?

A. It means modifying the feature screen to show the alternatives that are available in the subfamily that was the result of the first search based on the first selection.

Q. And so going back now to Exhibit A of 1028, what is the feature screen of this screen shot?

A. That would be the left section of that screen.

MR. TOMASULO: Jack, with your permission, could he just circle that on his exhibit?

MR. BALDWIN: Sure.

MR. TOMASULO: May I give him this pen, please?

A. (Witness complies.)

Q. Thank you.

So to let the record reflect -- now did you then draw a box around what you believed to be the feature screen?

A. Yes, which is the left part of the screen.

Q. Okay.

A. Which is visible partially here.

Q. I understand. So it's a portion of the feature screen?

A. Yes.

Q. Is there any -- but the feature screen does not -- does not include the parts that you haven't circled on at least this snapshot. So you've circled --

on Exhibit A, which is one page, you've circled every part of the feature screen on Exhibit A. Is that correct?

A. I need to look at the rest of the screen, but it looks like it shows the list of camera -- a list of cameras, and I don't know in which order this list is made, as well as the area that I circled that contains the feature names, like the brand and the list of brands, which are the alternatives, and the megapixels and the list of megapixels -- partial list because it's just a screen shot.

Q. So my question is even accepting that this screen shot may not be a complete screen shot, there's nothing on Exhibit A that is included in the feature screen other than what you've circled on this depiction?

A. Yes. At first view, yes.

Q. Okay. So, for instance, where it says -- it has a picture of a camera that says, Canon EOS Rebel XTi, that's not part of the feature screen?

A. No.

Q. Okay. And then up there where it has an ad for XM Radio, that's not part of the feature screen. Right?

A. Right.

Q. And everything above that ad for XM Radio, that's not part of the feature screen either. Right?

A. Right.

Q. Okay. And then so if we go to the next page of Exhibit 1028, we see Exhibit B.

A. Yes.

Q. Do you see that?

A. Yes.

Q. Okay. And then if we -- again, this shows the -- a more complete feature screen on the left side. Correct?

A. Yeah. It shows more -- more features and their alternatives, yes.

Q. I'll give you your pen back, please. Could you circle what the feature screen is on Exhibit B?

A. (Witness complies.)

Q. You can keep that.

A. Okay.

Q. So then the revision of the feature screen then requires a revision beyond simply indicating what's been selected. Correct?

A. Yes.

Q. -- and you take a look at step 1 K, revising said feature screen to indicate the available alternatives of the second subfamily, what does that mean in the context of this example?

A. It means displaying to the user a second -- a feature screen that indicates those available alternatives found in step J.

Q. So a better question: How did you learn about web servers?

A. We learned about web servers back in second quarter of '94 by attending, for example, seminars. I remember attending a seminar at Stanford University that explained how they worked and where to find one to play with and test.

Q. You said we attended --

A. Kris Kimbrough and I.

Q. You and Kris?

A. Yeah.

Q. Anyone else? At this time it would have been Danish International.

A. At the time in 1994?

Q. Yeah, '94, '95.

A. 1994 would have been just Kris and I.

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

```
                                  )
PARTSRIVER, INC.,                 )
   Plaintiff,,                    )
                                  )
v.                                )
                                  ) CASE NO. 2-07-CV-440 DF
SHOPZILLA, INC., YAHOO!           )
INC.; PRICEGRABBER.COM,           )
INC.; EBAY, INC.; and             )
MICROSOFT CORPORATION,            )
   Defendants.                    )
```

REPORTER'S CERTIFICATION
DEPOSITION OF SHERIF DANISH, VOLUME 2
JANUARY 21, 2009

I, Julie C. Brandt, Certified Shorthand Reporter in and for the State of Texas, hereby certify to the following:

That the witness, SHERIF DANISH, was duly sworn by the officer and that the transcript of the oral deposition is a true record of the testimony given by the witness;

That the deposition transcript was submitted on ____________________ to the witness or to the attorney for the witness for examination, signature and return to Merrill Legal Solutions by ____________________;

That the amount of time used by each party at the deposition is as follows:
MR. ZEMBEK.....00 HOUR(S):46 MINUTE(S)
MR. CEDEROTH.....02 HOUR(S):31 MINUTE(S)
MR. TOMASULO.....01 HOUR(S):36 MINUTE(S)
MS. DeRIEUX.....00 HOUR(S):00 MINUTE(S)
MR. BALDWIN.....00 HOUR(S):00 MINUTE(S)

That pursuant to information given to the deposition officer at the time said testimony was taken, the following includes counsel for all parties of record:

FOR THE PLAINTIFF:

Robert W. Coykendall
MORRIS LAING
300 N. Mead
Suite 200
Wichita, Kansas 67202-2722
316.262.2671
316.262.6226 (fax)
rcoykendall@morrislaing.com

Jack Baldwin
BALDWIN & BALDWIN, LLP
400 West Houston Street
Marshall, Texas 75670
903.935.4131
903.935.1397 (fax)
jbb@baldwinlaw.com

FOR THE DEFENDANTS MICROSOFT and eBAY:

Richard A. Cederoth
SIDLEY AUSTIN, LLP
One South Dearborn
Chicago, Illinois 60603
312.853.7026

312.853.7036 (fax)
rcederoth@sidley.com

Theodore W. Chandler
SIDLEY AUSTIN, LLP
555 West Fifth Street
Suite 4000
Los Angeles, California 90013
213.896.5830
213.896.6600 (fax)
tchandler@sidley.com

FOR THE DEFENDANT PRICEGRABBER.COM:

Michael A. Tomasulo
JONES DAY
555 South Flower Street
Fiftieth Floor
Los Angeles, California 90071
213.243.2619
213.243.2539 (fax)
matomasulo@jonesday.com

FOR THE DEFENDANT SHOPZILLA, INC.:

Elizabeth L. DeRieux
CAPSHAW DeRIEUX, LLP
1127 Judson Road
Suite 220
Longview, Texas 75601
903.236.9800
903.236.8787 (fax)
ederieux@capshawlaw.com

FOR THE DEFENDANT YAHOO! INC.:

Richard Zembek
FULBRIGHT & JAWORSKI
1301 McKinney
Suite 5100
Houston, Texas 77010-3095
713.651.5203
713.651.5246 (fax)
rzembek@fulbright.com

L. Joni Collins
FULBRIGHT & JAWORSKI
2200 Ross Avenue
Suite 2800
Dallas, Texas 75201-2784
214.855.8049
214.855.8200 (fax)

That $__________ is the deposition officer's charges to the Defendant Yahoo! Inc. for preparing the original deposition transcript and any copies of exhibits;

I further certify that I am neither counsel for, related to, nor employed by any of the parties or attorneys in the action in which this proceeding was taken, and further that I am not financially or otherwise interested in the outcome of the action.

Certified to by me January 29th, 2009.

Julie Brandt
Julie C. Brandt, CSR, CRR
Texas CSR No. 4018
Expiration Date: 12/31/10

Merrill Legal Solutions
Reg. No. 191
4144 North Central Expressway
Suite 850
Dallas, Texas 75204
800-966-4567

Exhibit 12

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

PARTSRIVER, INC.,)
Plaintiff,)
vs.)
) **Case No. 2-07CV-440 DF**
SHOPZILLA, INC.; VALUECLICK, INC.;)
PRICERUNNER LIMITED; YAHOO!, INC.;)
PRICEGRABBER.COM, INC.; EBAY, INC.;)
and MICROSOFT CORPORATION,)
Defendants.)
)

PLAINTIFF'S RESPONSES TO DEFENDANTS' FIRST SET OF JOINT INTERROGATORIES TO PARTSRIVER, INC.

Plaintiff in the above captioned matter, by and through its undersigned counsel, and pursuant to Rules 26 and 33 of the Federal Rules of Civil Procedure, hereby submit its responses to Defendants, Shopzilla, Inc., Valueclick, Inc., PriceRunner, Ltd., Yahoo!, Inc., PriceGrabber.com, Inc., eBay, Inc., and Microsoft Corporation's First Set of Interrogatories to Plaintiff.

INTERROGATORY NO. 1.:

Describe in detail the facts and circumstances of the conception of the alleged invention of each asserted claim of the '821 patent, including but not limited to the date of conception, the identity of all persons contributing to, involved with or witnessing such conception, and the identity of all documents relating to such conception.

RESPONSE:

In 1990-91 Sherif Danish became aware of problems AMP customers were facing, including:

EXHIBIT NO. 1015
Julie Brandt, CSR-RMR-CRR

1. Identifying part numbers for the numerous products offered by AMP; and
2. Difficulties in understanding and using the existing AMP paper catalogs, which were difficult to understand, with many products having a very similar look with only slight differences that were hard to distinguish.

A computerized solution seemed plausible, but existing search technology had problems with returning responses of "sorry no matches," when multiple attributes were selected.

Sherif Danish knew that a computerized solution could address these issues so he conceived of a method to narrow selection of items that satisfied selected criteria in a step-by-step manner. Based upon this idea, Kris Kimbrough and Sherif Danish worked together to design and implement a system to perform this method. Between February and April 1992, the methods described in claims 1 and 2 of the '821 patent were implemented as a demonstration program.

The conception and reduction to practice happened at Danish International, which is small company that has not retained documents from that period of time. Any potentially relevant computerized records for Danish International were lost when a computer was stolen on September 26, 2000.

Kris Kimbrough and Sherif Danish were the only persons involved with the conception of the method described in claims 1 and 2 of this patent. A demonstration program was created by April 1992, which is the only document known to exist that relates to the conception of the method shown in claims 1 and 2 of the '821 patent. A version of this demonstration program is submitted herewith.

<u>INTERROGATORY NO. 2:</u>:

Describe in detail the facts and circumstances of the first reduction to practice of the alleged invention of each asserted claim of the '821 patent, including but not limited to the date of such reduction to practice, the identity of all persons contributing to, involved with or witnessing such reduction to practice, and the identity of all documents relating to such reduction to practice.

RESPONSE:

The first two claims of patent '821 were first reduced to practice in April 1992, when a demonstration software program was written that utilized a limited number of AMP products. Kris Kimbrough and Sherif Danish were the persons involved in the process, and are the witnesses to the reduction to practice.

The conception and reduction to practice happened at Danish International, which is small company that has not retained documents from that period of time. Any potentially relevant computerized records for Danish International were lost when a computer was stolen on September 26, 2000.

The documents relevant to the reduction to practice that still exist are a software version of the demonstrations program, and the affidavits of Sherif Danish and Kris Kimbrough with attached Exhibits A-C, which were filed with the patent office and produced as document Nos. PR 0000236-245 and PR 0000252-264.

INTERROGATORY NO. 3:

Describe in detail the facts and circumstances relating to any alleged diligence between the conception and the first reduction to practice of the alleged invention of each asserted claim of the '821 patent, including but not limited to the date of such diligence, the identity of all persons contributing to, involved with or witnessing such diligence, and the identity of all documents relating to such diligence.

RESPONSE:

Between early 1992 and April 1992, Kris Kimbrough and Sherif Danish worked to structure a search process that achieved the goals of avoiding the limitations of the existing process. In an iterative collaboration, the structure of the method emerged, and, the feature screen, the method of selecting alternatives, and the way in which the search terms were combined to search over the entire data structure were developed. Kris Kimbrough and Sherif Danish were involved in this process. Kris Kimbrough, Amy Kimbrough, Magda Danish and Sherif Danish are witnesses to the diligence with which it was done.

INTERROGATORY NO. 4:

Identify all assignments, licenses, sublicenses, covenants not to sue, development agreements or other written or oral agreements (collectively "agreements") relating to the '821 patent, and for each of such agreement, identify its title, the parties, the effective date and whether the agreement is currently in effect.

RESPONSE:

There have been three assignments of this patent:

1. Assignment Number 1 was recorded on June 14, 2004, between Saqqara Systems, Inc, assignor and Co-Invest 2000 Fund, L.P., as Administrative Agent for assignees;
2. Assignment Number 2 was recorded on March 31, 2006, between Co-Invest 2000 Fund, L.P., as Administrative Agent for assignees, as assignor and Saqqara Systems, Inc, assignee; and
3. Assignment Number 3 was recorded on April 26, 2006, between Saqqara Systems, Inc, assignor and PartsRiver, Inc. Assignee.

All customers of Saqqara and PartsRiver who required the right to use ProductServer or Commerce Suite from 2001 forward acquired limited rights to practice the patent. See list attached.

INTERROGATORY NO. 5:

Identify the complete factual basis for PartsRiver's allegation that it owns all right, title and interest in the '821 patent and has standing to sue, including the identity of all documents related to such allegation and a description of all transfers of legal title.

RESPONSE:

See United States Patent and Trademark Office Notice of Recordation of Assignment Document, previously produced on February 25, 2008, and bates labeled PR0002608-2611; and the Asset Purchase Agreement previously produced on February 25, 2008, and bates labeled PR0002514-2586.

INTERROGATORY NO. 6:

Identify all persons or entities which held any rights in the '821 Patent as of the date this action was filed and the nature of the rights so held.

RESPONSE:

PartsRiver, Inc. and all customers identified in Response to Interrogatory No., 4, above. PartsRiver, Inc., owned all right, title and interest in and to the '821 patent. All of the customers identified have rights to practice the patent.

INTERROGATORY NO. 7:

Describe in detail the facts and circumstances relating to when each Defendant was provided with notice of (i) the '821 Patent and (ii) the Defendant's alleged infringement of the '821 Patent.

RESPONSE:

Subject to proof of earlier actual notice PartsRiver Inc., has information and believes that Defendants received notice of the '821 Patent upon its issuance and learned of their infringement of the '821 Patent upon service of the Complaint in this matter.

Respectfully submitted,

[illegible]

Robert W. Coykendall, SC#10137
MORRIS, LAING, EVANS, BROCK
& KENNEDY, CHARTERED
300 N. Mead, Suite 200
Wichita, Kansas 67202
Phone: (316) 262-2671
Facsimile: (316) 262-5991

Attorneys for Plaintiff

DECLARATION

The undersigned, under penalty of perjury, hereby declares that he is an agent of PartsRiver, Inc., and is authorized by the Company to answer the interrogatories, and the foregoing answers to interrogatories are true and correct to the best of his knowledge and belief, based upon his personal knowledge and the books and records of PartsRiver, Inc.

[illegible]

Sherif Danish

Original Buy Date	NAME	Latest Product Used	Type	Comments
5/23/2000	3M	CommerceSuite 4.5	Licensed	current
4/15/2000	Honeywell International, Inc.	CommerceSuite 4.5	Licensed	current
4/5/2000	Hubbell	ProductServer 3.1	Licensed	current
7/31/2000	IDEC Corporation	CommerceSuite 4.5	Licensed	current
3/30/2001	Lincoln Electric Company	CommerceSuite 4.5	Licensed	current
5/31/2002	SICK, Inc.	CommerceSuite 4.5	Licensed	current
12/29/1999	Zilog	ProductServer 3.1	Licensed	current
9/14/2003	Cooper Industries	CDM	Hosting	current
6/28/2000	CRC Industries	CommerceSuite 4.0.2	Licensed	expired
2/14/2001	Eastman Chemical Company	CommerceSuite 4.5	Licensed	expired
8/18/1999	Eaton Corporation	CommerceSuite 4.5	Licensed	expired
7/28/2000	Fairchild	ProductServer 3.1	Hosting	expired
6/28/2001	FCI, Inc.	CommerceSuite 4.5	Licensed	expired
4/26/2000	GE Industrial Systems	ProductServer 3.1	Licensed	expired
12/31/2002	Hagemeyer	ProductServer 3.1	Licensed	expired
10/31/2001	HBD Industries	ProductServer 3.1	Licensed	expired
9/10/1999	Maxim	ProductServer 2.2	Licensed	expired
10/1/1999	McMaster Carr	ProductServer 3.1	Licensed	expired
12/22/2000	Memec	ProductServer 3.1	Licensed	expired
11/18/1999	Micron Technology	CommerceSuite 4.5	Licensed	expired
12/29/1999	Molex Incorporated	CommerceSuite 4.5	Licensed	expired
10/31/2000	National Starch & Chemical	ProductServer 3.1	Licensed	expired
6/28/2000	New Age Industries	ProductServer 3.1	Licensed	expired
9/29/2000	Philips Logic	ProductServer 2.01	Hosting	expired
9/14/2000	Radio Frequency Systems	ProductServer 3.1	Licensed	expired
7/5/2000	Siemon Company	ProductServer 3.1	Licensed	expired
3/31/2000	SiliconExpert	ProductServer 3.1	Licensed	expired
3/31/2000	Spectrum Control Inc.	ProductServer 3.1	Licensed	expired
1/7/2004	Telcordia Technologies	CommerseSuite 4.5	Licensed	expired
10/1/2002	Thomas Technology Solutions	CommerceSuite 4.5	Licensed	expired
2/14/2001	Weidmuller Inc	CommerceSuite 4.0	Licensed	expired
12/20/2000	Waters Technology	ProductServer 3.1	Licensed	expired

Exhibit 13

EXHIBIT NO. 1028

Julie Brandt, CSR-RMR-CRR

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

PARTS RIVER, INC.	)	
Plaintiff,	)	
vs.	)	**Case No. 2-07CV-440 DF**
	)	
SHOPZILLA, INC.; VALUECLICK, INC.;	)	
PRICERUNNER, LTD.; YAHOO!, INC.;	)	**JURY**
PRICEGRABBER.COM, INC.; EBAY, INC.;	)	
and MICROSOFT CORPORATION	)	
Defendants.	)	

<u>PLAINTIFF'S DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS</u>

The Plaintiff, PARTSRIVER, Inc. ("PartsRiver") pursuant to Rule 3-1, hereby provides its Disclosure of Asserted Claims and Infringement Contentions. These preliminary contentions are applicable to U.S. Patent 6,275,821 ("'821 Patent").

(A) PartsRiver contends that the defendants have infringed and continue to infringe claims 1 and 2 of the '821 Patent.

(B) PartsRiver contends that the defendants exercise control over and operate their susidiaries', partners and foreign websites which utilize search processes or methods which directly infringe and induce infringement of Claims 1 and 2 of the '821 patent.

(C) PartsRiver contends that the following websites operated by the defendants utilize search processes or methods which infringe upon both Claims 1 and 2 of the '821 patent:

Shopzilla, Inc.:	www.shopzilla.com
	www.shopzilla.co.uk
	www.shopzilla.fr/
	www.shopzilla.de/

	www.bizrate.com
	www.bizrate.co.uk
	http://fr.bizrate.com
	www.bizrate.de
Valueclick, Inc./Pricerunner, Ltd.:	www.pricerunner.com
	www.pricerunner.co.uk
	www.pricerunner.fr
	www.pricerunner.de
	www.pricerunner.at
	www.pricerunner.se
	www.pricerunner.dk
Yahoo!, Inc.:	shopping.yahoo.com
	yahoo.buscape.com.br
	yahoo.buscape.com.ar
	http://yahoo.shoptoit.ca/
	http://shopping.yahoo.co.uk
	http://shopping.yahoo.com.au/
	http://shopping.yahoo.dk/
	http://shopping.yahoo.no/
	http://shopping.yahoo.fr/
	http://shopping.yahoo.es/

http://shopping.yahoo.nl/

http://shopping.yahoo.it

http://shopping.yahoo.se

autos.yahoo.com

hotjobs.yahoo.com

http://yahoo.mycareer.com.au/

Pricegrabber.com, Inc.: www.pricegrabber.com

http://www.preciomania.com

http://mx.preciomania.com

www.pricegrabber.ca

www.pricegrabber.co.uk

ask2.pricegrabber.com

http://www.about.com/shopping/

Abesofmaine.pgpartner.com

Amdzone.pgpartner.com

arstechnica.pgpartner.com

cameralabs.pgpartner.com

cheapstingybargains.pgpartner.com

dealdetectives.pgpartner.com

dealsdigger.pgpartner.com

dealnews.pgpartner.com

dealsofamerica.pgpartner.com

dealplus.pgpartner.com

dealsplus.pgpartner.com

edealinfo.pgpartner.com

firstglimpsemag.pgpartner.com

geardigest.pgpartner.com

gizmosforgeeks.pgpartner.com

hdtvsolutions.pgpartner.com

hipcheq.pgpartner.com

judysbook.pgpartner.com

justrelevant.pgpartner.com

legitreviews.pgpartner.com

logicbuy.pgpartner.com

markdownmonkey.pgpartner.com

	modemhelp.pgpartner.com
	cmodshop.pgpartner.com
	parenthood.pgpartner.com
	pdablast.pgpartner.com
	printerinfo.pgpartner.com
	steves-digicams.pgpartner.com
	thenerds.pgpartner.com
	twitchguru.pgpartner.com
	usatoday.pgpartner.com
Ebay, Inc.:	www.shopping.com
	http://fr.shopping.com
	http://de.shopping.com
	http://uk.shopping.com
	http://au.shopping.com
	www.dealtime.com
	www.dealtime.co.uk
	www.ugenie.com
	www.epinions.com

www.pricetool.com

www.express.ebay.com

Microsoft Corporation: shopping.msn.com

http://shopping.sympatico.msn.ca/

http://magasiner.sympatico.msn.ca/

http://shopping.msn.co.jp/

http://shopping.ninemsn.com.au/

http://shopping.msn.fr/

http://shopping.msn.nl/

http://shopping.msn.de/

http://shopping.msn.co.uk/

(D) Attached hereto are charts identifying specifically where each element of Claims 1 and 2 of the '821 Patent is found within at least one website for each defendant identified above. The other listed websites infringe in the same manner as shown in the claims chart(s) for a defendant's website. The charts are based upon pages as they existed in January and February, 2008, and given the nature of the sites, and the information displayed on those sites, the information that is retrieved from the application of the method as shown in the charts will change.

(E) It is claimed that each of the claim elements is literally infringed.

(F) Claims 1 and 2 of the '821 Patent claim priority to October 14, 1994.

(G) PartsRiver preserves the right to rely on the assertion that its Product Server software product and Commerce Suite software product practice the claimed invention. Attached hereto are charts identifying specifically with examples where the products incorporate or reflect each of the elements of Claims 1 and 2.

Respectfully submitted,

Jack Baldwin, Texas State Bar No. 01625330
BALDWIN & BALDWIN, L.L.P.
400 West Houston Street
P. O. Drawer 1349
Marshall, Texas 75671
Phone: (903) 935-4131
Facsimile: (903) 935-9538

and

[illegible]
Kerr M. Peterson, Kansas State Bar No. 07499
Robert W. Coykendall, Kansas State Bar No. 10137
MORRIS, LAING, EVANS, BROCK
& KENNEDY, CHARTERED
200 West Douglas, 4th Floor
Wichita, Kansas 67202-3084
Phone: (316) 262-2671
Facsimile: (316) 262-5991

ATTORNEYS FOR PLAINTIFF

CERTIFICATE OF SERVICE

The undersigned hereby certifies that on February 18, 2008, pursuant to Patent Rule 3-1, the Disclosure of Asserted Claims and Preliminary Infringement Contentions were served on the defendants via electronic mail.

[illegible]

Robert W. Coykendall

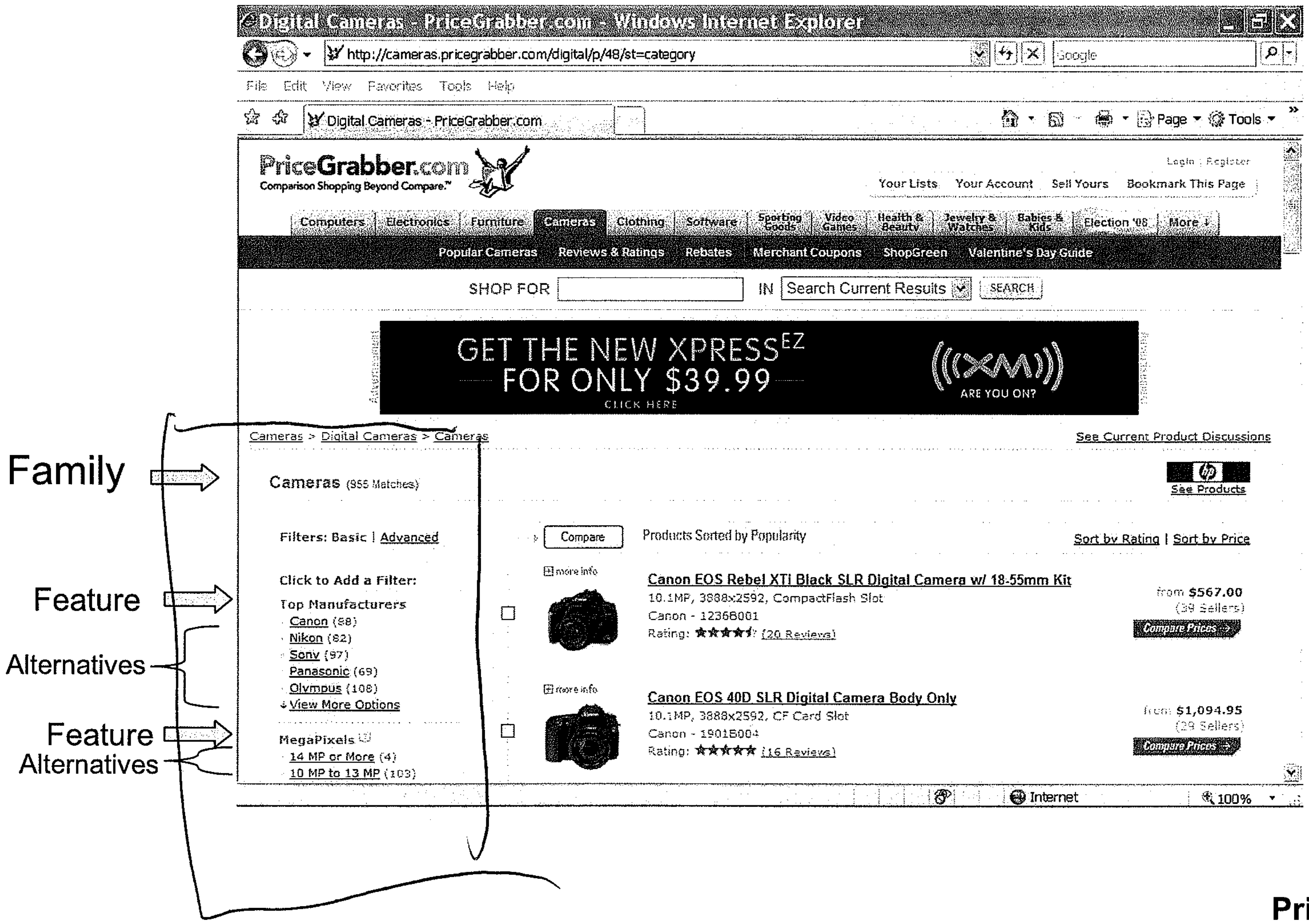
Elements of Claims 1 and 2 Patent 6,275,821	**Location of Elements in Pricegrabber.com Website**
1. A method for assisting a user in identifying a subfamily of items within a family of items comprising the steps of:	
(a) providing a computer readable data file of stored information representing at least one family of items, said data file identifying at least one alternative for each item,	Pricegrabber.com accesses a data file of stored information containing multiple families of items and identifying at least one alternative for each item. See "Shop by Category" at http://www.Pricegrabber.com/, which identifies families or groups of families of products. For example, one family of items within the group of "Cameras" is the "Digital Cameras" family. If "Digital Cameras" is selected, the following page is displayed. http://cameras.pricegrabber.com/digital/c/76/st=category From there, the link for "cameras" under "Browse Cameras" sends the user to the "Digital Cameras" feature screen.
(b) reading said data file,	When the link for "Cameras" is selected, Pricegrabber.com accesses and reads the data file.
(c) displaying a feature screen indicating said alternatives represented in the family,	Pricegrabber.com displays a feature screen, in this example, for the digital camera family, attached Exhibit "A": http://cameras.pricegrabber.com/digital/p/48/st=category

(d) accepting a first selection criteria of at least one alternative	When an alternative is selected, Pricegrabber.com accepts the first selection criteria of at least one alternative. See, for example, attached Exhibits "A" and "B". From the main feature screen for digital cameras (Exhibit "A"), the user moves the cursor over the link for the "alternative," and the status bar shows the link for the search for that alternative. For example, when the curser is on the alternative Optical Zoom "7x to 10x", the status bar shows the URL: http://cameras.pricegrabber.com/digital/p/48/st=category/popup3[]=104:974 (Exhibit "B"). When the user clicks on the alternative to select it, Pricegrabber.com accepts the selection criteria.
(e) determining a first subfamily of items wherein each said item in the first subfamily satisfies said first selection criteria,	Pricegrabber.com determines which digital cameras meet the "selection criteria" of Optical Zoom "7x to 10x"; these items comprise the "subfamily."
(f) determining available alternatives represented in the first subfamily,	Pricegrabber.com determines what alternatives are available within the "subfamily" of digital cameras satisfying the first selection criteria, i.e., available alternatives within the subfamily of digital cameras with Optical Zoom "7x to 10x". In this example, it finds that 46 items are within that subfamily. See, attached Exhibit "C".
(g) revising said feature screen to indicate the available alternatives of the first subfamily,	Pricegrabber.com revises the feature screen to indicate available alternatives for the subfamily. See attached Exhibit "C": http://cameras.pricegrabber.com/digital/p/48/st=category/popup3[]=104:974

(h) accepting a second selection criteria comprising the alternative or alternatives of the first selection criteria plus at least one alternative selected from the revised feature screen,	When a user chooses one of the available alternatives for the subfamily, Pricegrabber.com accepts a second selection criteria comprising the alternatives of the first selection criteria (i.e., Optical Zoom "7x to 10x") plus at least one alternative selected from the revised feature screen. See attached Exhibits "C" and "D." For example, the user can make a second selection, i.e., Megapixels "7 MP to 9 MP". When the user places the cursor over the link for that alternative, the status bar shows: http://cameras.pricegrabber.com/digital/p/48/st=category/popup3[]=104:974/popup1[]=90:975 (Exhibit "D"). When the link is selected by the user, Pricegrabber.com "accepts" the selection criteria which consists of the first selection criteria (Optical Zoom "7x to 10x") and "7 MP to 9 MP".
(i) determining a second subfamily of items of the family wherein each item in the second subfamily satisfies said second selection criteria,	Pricegrabber.com determines which digital cameras meet the "selection criteria" of "7 MP to 9 MP" <u>and</u> Optical Zoom "7x to 10x"; these items comprise a "second subfamily."
(j) determining available alternatives represented in the second subfamily, and	Pricegrabber.com determines what alternatives are available within the second subfamily of items.
(k) revising said feature screen to indicate the available alternatives of the second subfamily.	Pricegrabber.com revises the "feature screen" that displays all alternatives available for digital cameras satisfying the alternative "7 MP to 9 MP" <u>and</u> Optical Zoom "7x to 10x". These items comprise a "second subfamily." See attached Exhibit "E": http://cameras.pricegrabber.com/digital/p/48/st=category/popup3[]=104:974/popup1[]=90:975 In this example, there are 22 products in the second subfamily.

2. The method of claim 1 wherein each family has at least one feature associated therewith and further comprising the step of:	Pricegrabber.com accesses a data file of stored information representing at least one family of items wherein each family has at least one feature associated therewith. In this example, the digital camera family has features including megapixels and top manufacturer.
displaying at least one grouping wherein each said grouping comprises one of said features visually related to respective alternatives.	See attached Exhibit "A": http://cameras.pricegrabber.com/digital/p/48/st=category The page shows alternatives grouped under each feature and separated from other features and associated alternatives. For example, the "Top Manufacturer" feature groups the available alternatives (including "Canon", "Nikon", etc.) as a list beneath the the feature heading "Top Manufacturer". These manufacturer alternatives are visually separated from alternatives "14 MP or more" and "10 MP to 13MP" under the megapixels feature.

Feature Screen for Digital Camera Family


Family
Feature
Alternatives
Feature
Alternatives
Digital Cameras - PriceGrabber.com - Windows Internet Explorer
http://cameras.pricegrabber.com/digital/p/48/st=category
File Edit View Favorites Tools Help
Digital Cameras - PriceGrabber.com
PriceGrabber.com
Comparison Shopping Beyond Compare.™
Login | Register
Your Lists Your Account Sell Yours Bookmark This Page
Computers Electronics Furniture Cameras Clothing Software Sporting Goods Video Games Health & Beauty Jewelry & Watches Babies & Kids Election '08 More
Popular Cameras Reviews & Ratings Rebates Merchant Coupons ShopGreen Valentine's Day Guide
SHOP FOR IN Search Current Results SEARCH
GET THE NEW XPRESS EZ FOR ONLY $39.99 CLICK HERE
ARE YOU ON?
See Current Product Discussions
See Products
Compare
Sort by Rating | Sort by Price
Compare Prices
Internet
100%

Cameras > Digital Cameras > Cameras

Cameras (955 Matches)

Filters: Basic | Advanced

Click to Add a Filter:

Top Manufacturers
- Canon (88)
- Nikon (82)
- Sony (97)
- Panasonic (69)
- Olympus (108)
- View More Options

MegaPixels
- 14 MP or More (4)
- 10 MP to 13 MP (103)

Products Sorted by Popularity

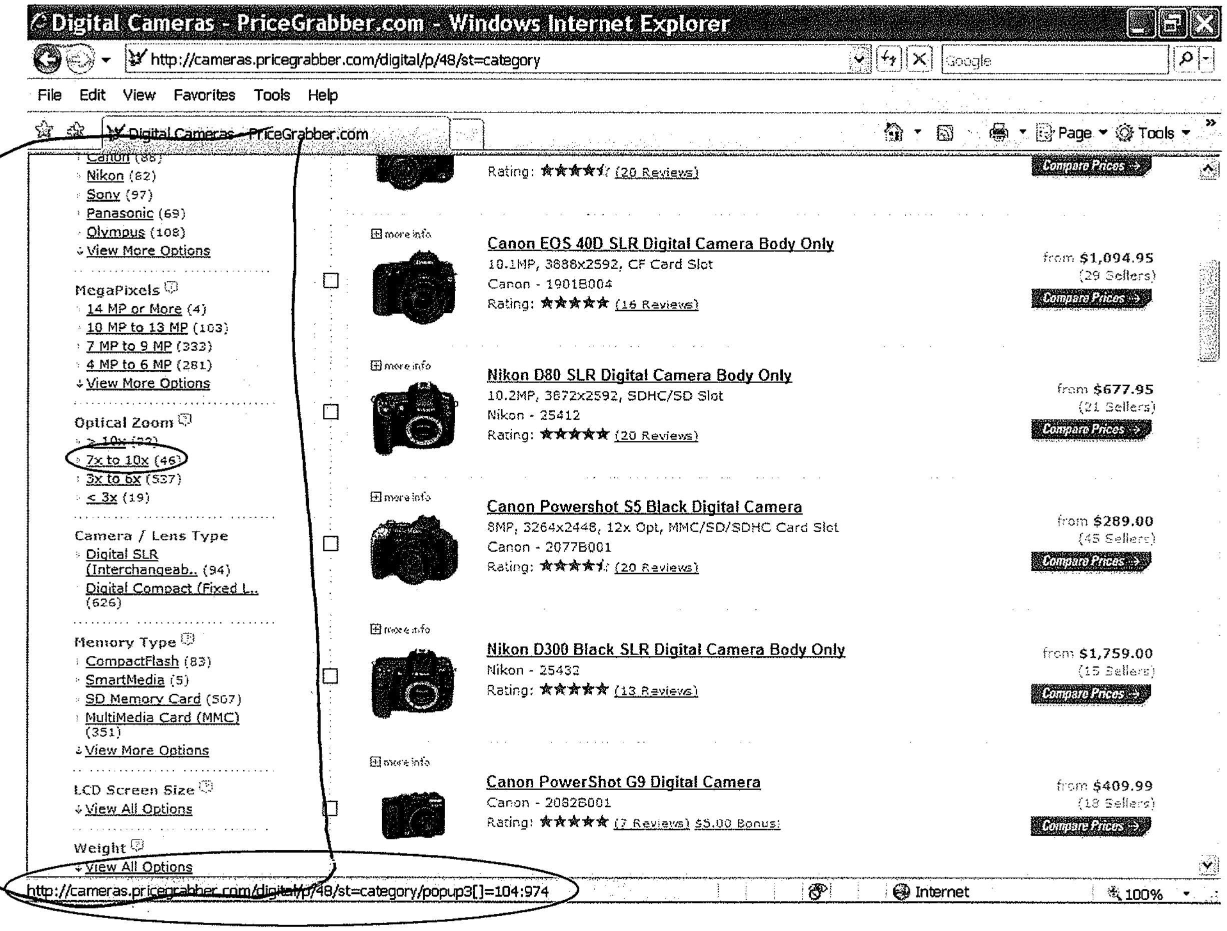
Canon EOS Rebel XTi Black SLR Digital Camera w/ 18-55mm Kit
10.1MP, 3888x2592, CompactFlash Slot
Canon - 12368001
Rating: ★★★★ (20 Reviews)
from $567.00 (39 Sellers)

Canon EOS 40D SLR Digital Camera Body Only
10.1MP, 3888x2592, CF Card Slot
Canon - 1901B004
Rating: ★★★★★ (16 Reviews)
from $1,094.95 (29 Sellers)

Exhibit A

Pricegrabber.com

Feature Screen with cursor on alternative Optical Zoom "7x to 10x"



Digital Cameras - PriceGrabber.com - Windows Internet Explorer

http://cameras.pricegrabber.com/digital/p/48/st=category

File Edit View Favorites Tools Help

Digital Cameras - PriceGrabber.com

Canon (86)
Nikon (82)
Sony (97)
Panasonic (69)
Olympus (108)
View More Options

MegaPixels
14 MP or More (4)
10 MP to 13 MP (103)
7 MP to 9 MP (333)
4 MP to 6 MP (281)
View More Options

Optical Zoom
≥ 10x (52)
7x to 10x (46)
3x to 6x (537)
< 3x (19)

Camera / Lens Type
Digital SLR (Interchangeab.. (94)
Digital Compact (Fixed L.. (626)

Memory Type
CompactFlash (83)
SmartMedia (5)
SD Memory Card (567)
MultiMedia Card (MMC) (351)
View More Options

LCD Screen Size
View All Options

Weight
View All Options

Rating: ★★★★☆ (20 Reviews) — Compare Prices

Canon EOS 40D SLR Digital Camera Body Only
10.1MP, 3888x2592, CF Card Slot
Canon - 1901B004
Rating: ★★★★★ (16 Reviews)
from $1,094.95 (29 Sellers) — Compare Prices

Nikon D80 SLR Digital Camera Body Only
10.2MP, 3872x2592, SDHC/SD Slot
Nikon - 25412
Rating: ★★★★★ (20 Reviews)
from $677.95 (21 Sellers) — Compare Prices

Canon Powershot S5 Black Digital Camera
8MP, 3264x2448, 12x Opt, MMC/SD/SDHC Card Slot
Canon - 2077B001
Rating: ★★★★☆ (20 Reviews)
from $289.00 (45 Sellers) — Compare Prices

Nikon D300 Black SLR Digital Camera Body Only
Nikon - 25432
Rating: ★★★★★ (13 Reviews)
from $1,759.00 (15 Sellers) — Compare Prices

Canon PowerShot G9 Digital Camera
Canon - 2082B001
Rating: ★★★★★ (7 Reviews) $5.00 Bonus!
from $409.99 (18 Sellers) — Compare Prices

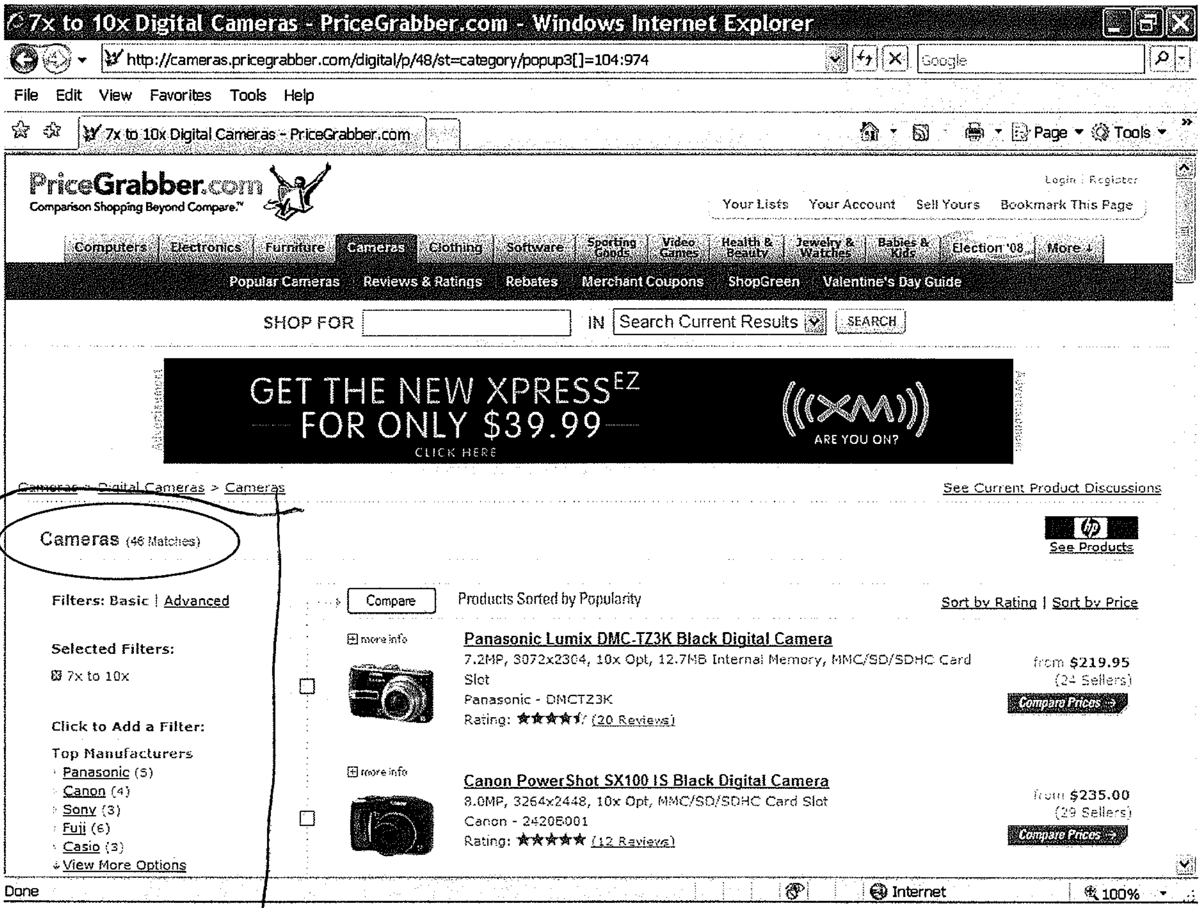
http://cameras.pricegrabber.com/digital/p/48/st=category/popup3[]=104:974
Internet 100%

Exhibit B

Pricegrabber.com

Revised feature screen: Subfamily=digital cameras with Optical Zoom "7x to 10x"

46 cameras are in the first subfamily



7x to 10x Digital Cameras - PriceGrabber.com - Windows Internet Explorer

http://cameras.pricegrabber.com/digital/p/48/st=category/popup3[]=104:974

File Edit View Favorites Tools Help

PriceGrabber.com Comparison Shopping Beyond Compare.™

Login | Register

Your Lists | Your Account | Sell Yours | Bookmark This Page

Computers | Electronics | Furniture | Cameras | Clothing | Software | Sporting Goods | Video Games | Health & Beauty | Jewelry & Watches | Babies & Kids | Election '08 | More

Popular Cameras | Reviews & Ratings | Rebates | Merchant Coupons | ShopGreen | Valentine's Day Guide

SHOP FOR [ ] IN Search Current Results SEARCH

GET THE NEW XPRESS EZ FOR ONLY $39.99 CLICK HERE (XM) ARE YOU ON?

Cameras > Digital Cameras > Cameras

See Current Product Discussions

Cameras (46 Matches)

See Products

Filters: Basic | Advanced

Selected Filters:
7x to 10x

Click to Add a Filter:

Top Manufacturers
- Panasonic (5)
- Canon (4)
- Sony (3)
- Fuji (6)
- Casio (3)
- View More Options

Compare | Products Sorted by Popularity | Sort by Rating | Sort by Price

Panasonic Lumix DMC-TZ3K Black Digital Camera
7.2MP, 3072x2304, 10x Opt, 12.7MB Internal Memory, MMC/SD/SDHC Card Slot
Panasonic - DMCTZ3K
Rating: ★★★★½ (20 Reviews)
from $219.95 (24 Sellers) Compare Prices

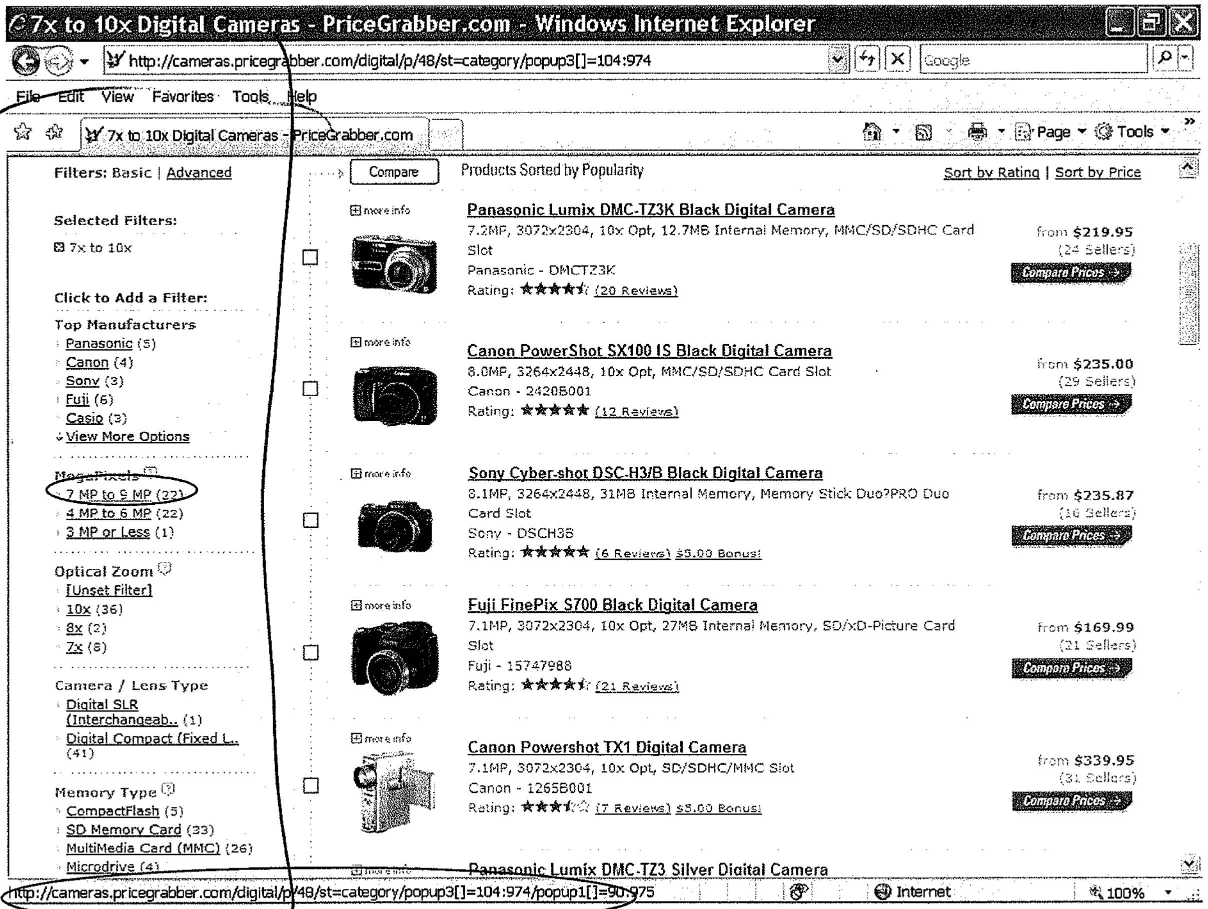
Canon PowerShot SX100 IS Black Digital Camera
8.0MP, 3264x2448, 10x Opt, MMC/SD/SDHC Card Slot
Canon - 2420B001
Rating: ★★★★★ (12 Reviews)
from $235.00 (29 Sellers) Compare Prices

Done | Internet | 100%

Exhibit C

Pricegrabber.com

Revised Feature Screen with cursor on alternative “7 MP to 9 MP””



7x to 10x Digital Cameras - PriceGrabber.com - Windows Internet Explorer
http://cameras.pricegrabber.com/digital/p/48/st=category/popup3[]=104:974
Google
File Edit View Favorites Tools Help
7x to 10x Digital Cameras - PriceGrabber.com
Page Tools

Filters: Basic | Advanced

Selected Filters:
7x to 10x

Click to Add a Filter:

Top Manufacturers
- Panasonic (5)
- Canon (4)
- Sony (3)
- Fuji (6)
- Casio (3)
- View More Options

MegaPixels
- 7 MP to 9 MP (22)
- 4 MP to 6 MP (22)
- 3 MP or Less (1)

Optical Zoom
- [Unset Filter]
- 10x (36)
- 8x (2)
- 7x (8)

Camera / Lens Type
- Digital SLR (Interchangeab.. (1)
- Digital Compact (Fixed L.. (41)

Memory Type
- CompactFlash (5)
- SD Memory Card (33)
- MultiMedia Card (MMC) (26)
- Microdrive (4)

Compare Products Sorted by Popularity Sort by Rating | Sort by Price

more info **Panasonic Lumix DMC-TZ3K Black Digital Camera**
7.2MP, 3072x2304, 10x Opt, 12.7MB Internal Memory, MMC/SD/SDHC Card Slot
Panasonic - DMCTZ3K
Rating: ★★★★½ (20 Reviews)
from $219.95 (24 Sellers) Compare Prices

more info **Canon PowerShot SX100 IS Black Digital Camera**
8.0MP, 3264x2448, 10x Opt, MMC/SD/SDHC Card Slot
Canon - 2420B001
Rating: ★★★★★ (12 Reviews)
from $235.00 (29 Sellers) Compare Prices

more info **Sony Cyber-shot DSC-H3/B Black Digital Camera**
8.1MP, 3264x2448, 31MB Internal Memory, Memory Stick Duo?PRO Duo Card Slot
Sony - DSCH3B
Rating: ★★★★★ (6 Reviews) $5.00 Bonus!
from $235.87 (16 Sellers) Compare Prices

more info **Fuji FinePix S700 Black Digital Camera**
7.1MP, 3072x2304, 10x Opt, 27MB Internal Memory, SD/xD-Picture Card Slot
Fuji - 15747988
Rating: ★★★★½ (21 Reviews)
from $169.99 (21 Sellers) Compare Prices

more info **Canon Powershot TX1 Digital Camera**
7.1MP, 3072x2304, 10x Opt, SD/SDHC/MMC Slot
Canon - 1265B001
Rating: ★★★½☆ (7 Reviews) $5.00 Bonus!
from $339.95 (31 Sellers) Compare Prices

more info **Panasonic Lumix DMC-TZ3 Silver Digital Camera**

http://cameras.pricegrabber.com/digital/p/48/st=category/popup3[]=104:974/popup1[]=90:975 Internet 100%

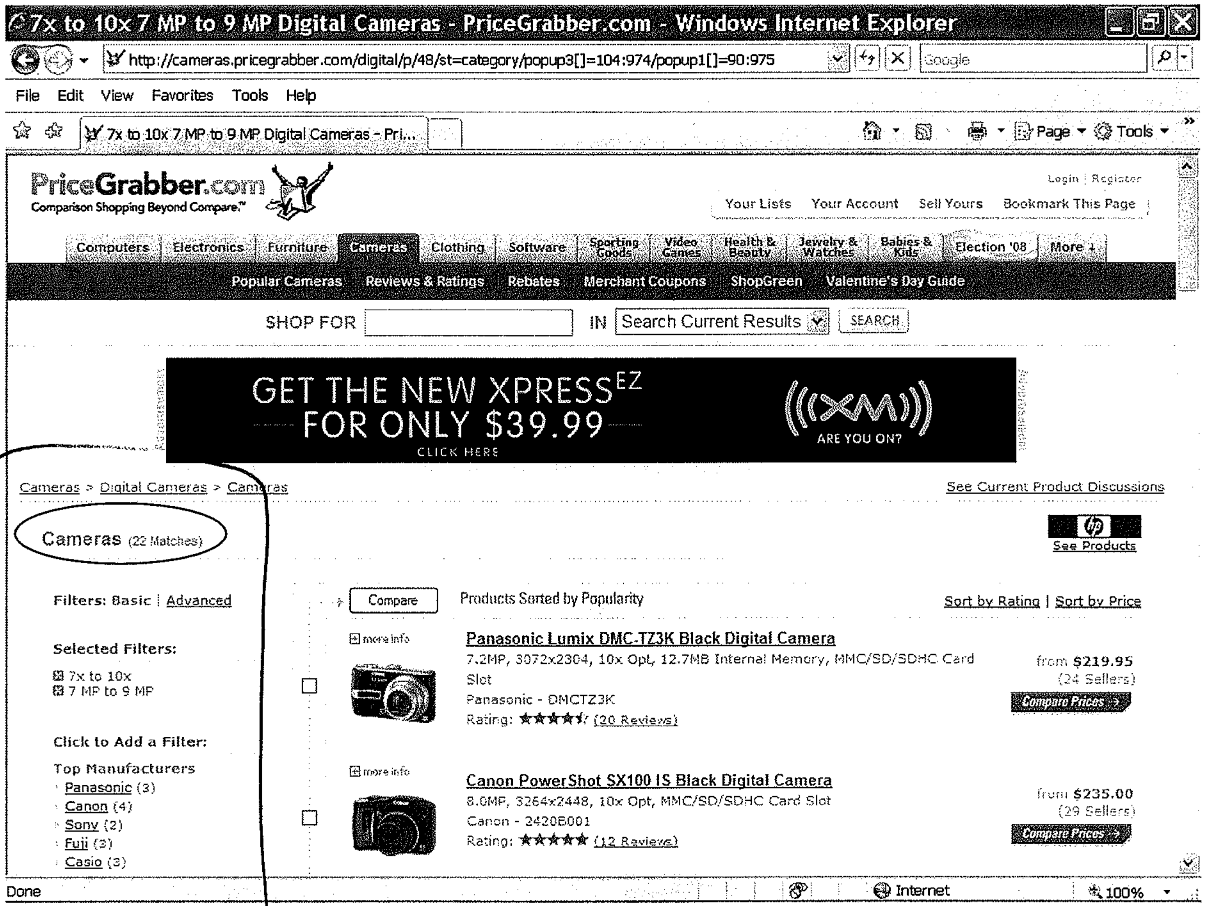
Status bar reflects the alternative “7 MP to 9 MP”

Exhibit D

Pricegrabber.com

Second revised feature screen displaying the second subfamily and available alternatives

22 cameras are in the second subfamily



7x to 10x 7 MP to 9 MP Digital Cameras - PriceGrabber.com - Windows Internet Explorer

http://cameras.pricegrabber.com/digital/p/48/st=category/popup3[]=104:974/popup1[]=90:975

File Edit View Favorites Tools Help

7x to 10x 7 MP to 9 MP Digital Cameras - Pri...

PriceGrabber.com
Comparison Shopping Beyond Compare.™

Login | Register

Your Lists | Your Account | Sell Yours | Bookmark This Page

Computers | Electronics | Furniture | Cameras | Clothing | Software | Sporting Goods | Video Games | Health & Beauty | Jewelry & Watches | Babies & Kids | Election '08 | More

Popular Cameras | Reviews & Ratings | Rebates | Merchant Coupons | ShopGreen | Valentine's Day Guide

SHOP FOR [ ] IN Search Current Results SEARCH

GET THE NEW XPRESS EZ FOR ONLY $39.99
CLICK HERE
((XM)) ARE YOU ON?

Cameras > Digital Cameras > Cameras

See Current Product Discussions

Cameras (22 Matches)

See Products

Filters: Basic | Advanced

Selected Filters:
- 7x to 10x
- 7 MP to 9 MP

Click to Add a Filter:

Top Manufacturers
- Panasonic (3)
- Canon (4)
- Sony (2)
- Fuji (3)
- Casio (3)

Compare | Products Sorted by Popularity | Sort by Rating | Sort by Price

Panasonic Lumix DMC-TZ3K Black Digital Camera
7.2MP, 3072x2304, 10x Opt, 12.7MB Internal Memory, MMC/SD/SDHC Card Slot
Panasonic - DMCTZ3K
Rating: ★★★★½ (20 Reviews)
from $219.95 (24 Sellers)
Compare Prices

Canon PowerShot SX100 IS Black Digital Camera
8.0MP, 3264x2448, 10x Opt, MMC/SD/SDHC Card Slot
Canon - 2420B001
Rating: ★★★★★ (12 Reviews)
from $235.00 (29 Sellers)
Compare Prices

Done | Internet | 100%

Exhibit E

Pricegrabber.com